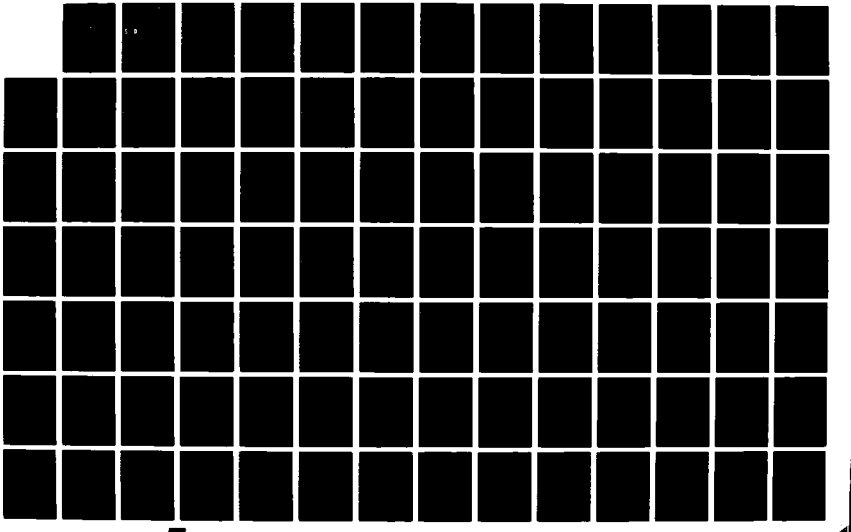
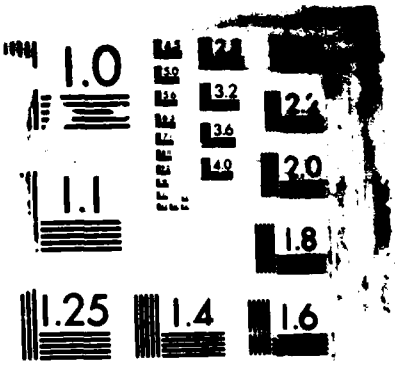
AD-A179 726 SPECTRAL FACTORIZATION AND HOMOGENIZATION METHODS FOR MODELING AND CONTRO..(U) SYSTEMS ENGINEERING INC GREENBELT MD W H BENNETT ET AL. 15 DEC 86 SEI-TR-86-14 1/3

UNCLASSIFIED AFOSR-TR-87-0502 F49620-84-C-0115 F/G 22/2 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
3.2
3.6
4.0

AFOSR-TR- 87-0562

Spectral Factorization and Homogenization Methods for Modeling and Control of Flexible Structures

Prepared by:

W.H. Bennett, G.L. Blankenship, H.G. Kwatny

SEI TR-86-14

Final Report
Sept. 1984 – Sept. 1986
AFOSR Contract No. F49620-84-C-0115

December 15, 1986

Submitted to:
AFOSR/NA
Bolling Air Force Base
Washington, DC 20332-6448
Attn: Dr. A. Amos

Submitted by:
SYSTEMS ENGINEERING, INC.
7833 Walker Drive – Suite 308
Greenbelt, MD 20770

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Air Force or the U.S. Government.

UNCLASSIFIED

CURITY CLASSIFICATION OF THIS PAGE

AD-A179726

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release, distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SEI TR-86-14	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 87-0502
6a. NAME OF PERFORMING ORGANIZATION Systems Engineering, Inc. — 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFOSR/NA
6c. ADDRESS (City, State and ZIP Code) 7833 Walker Dr. - Suite 308 Greenbelt, MD 20770	7b. ADDRESS (City, State and ZIP Code) Bolling Air Force Base Washington, D.C. 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR/NA — 8b. OFFICE SYMBOL (If applicable) NA	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F49620-84-C-0115
8c. ADDRESS (City, State and ZIP Code) Bolling Air Force Base Washington, D.C. 20332-6448	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. 61102F — PROJECT NO. 3005 — TASK NO. A1 — WORK UNIT NO.
11. TITLE (Include Security Classification) Spectral Factorization and Homogenization Methods	
12. PERSONAL AUTHOR(S) William Bennett, G.L. Blankenship and H.G. Kwatny	
13a. TYPE OF REPORT Final — 13b. TIME COVERED FROM Sept. 1984 TO Sept 86	14. DATE OF REPORT (Yr., Mo., Day) 1986, December 15 — 15. PAGE COUNT
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Distributed Parameter Control, homogenization, flexible structure control continuum modeling of structures

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

See the back side of this sheet.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ OTIC USERS ☐	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Anthony Amos	22b TELEPHONE NUMBER (Include Area Code) 202-767-4937 — 22c. OFFICE SYMBOL AFOSR/NA

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

19.ABSTRACT

This report describes the results of a two year research project on continuum modeling and vibration control of flexible structures with application to active control of vibrations in large space structures. A comprehensive methodology is discussed for the construction of effective (linear) models for large composite structures consisting of various flexible members (e.g. beams, trusses, etc.) and rigid body elements. Since our ultimate concern is the active, feedback control of such systems, we find it convenient to concentrate on frequency domain modeling. We start at the component level and show a systematic procedure for computing the irrational transfer functions required. Then by standard transform methods a complete *hybrid* model is developed. The methods were coded in a computer algebra system (SMP was used) which automated the model building process and produced Fortran code for numerical evaluation of the frequency responses.

Under certain conditions the dynamics of such large, low mass structures, having a regular infrastructure, can be modeled by the dynamics of continuua; e.g., trusses can be modeled as elastic beams. We demonstrate how effective continuum models of lattice structures with regular infrastructure can be obtained by a systematic procedure based on an asymptotic analysis of multiple scales called *homogenization*. This method is applied to several examples and it is shown that accurate computation of the required parameters of such continuum models is somewhat more subtle than merely averaging over lattice cells. Certain "corrector" formulae, required for such computations, are derived for several examples. In addition the nature of the process of combining homogenization and optimal control is discussed in detail.

For the computation of distributed parameter control we concentrated on an optimal frequency domain method based on solving an associated Wiener-Hopf problem. The method which was popularized by Jon Davis employs effective numerical algorithms (e.g. FFT, etc.) to compute a certain *spectral factorization* of a possibly matrix-valued (in the multiple control case) Hermittian, positive-definite transform by sampling the frequency response. The control laws considered in this report take the form of *distributed state feedback* with respect to a naturally defined, distributed state-space of functions over the spatial domain of the structure.

Several examples are considered in detail and certain numerical sensitivities are noted for the method. The relationship between this method for distributed control and more standard methods based on finite-dimensional, reduced-order modeling is discussed. In this context the distributed parameter method raises several basic questions with respect to continuum modeling which never arise when finite-dimensional approximations are used from the outset. In particular, as discussed by Gibson, the nature of various standard models for structural damping is deemed crucial.

Our ultimate goal in this project was to demonstrate both effective methods for continuum modeling of low mass density structures and computation of stabilizing, optimal control laws for these systems. Examples indicate that stabilization *and* control of essentially *all* controllable modes of the given system will be difficult to achieve in practice using a finite array of sensors. However, it is shown that, under certain conditions, an aribitrary, finite number of modes can be controlled using these methods *without destabilizing* the remaining uncontrolled modes.

Foreward

This research was performed by SYSTEMS ENGINEERING, INC., Greenbelt, MD under contract No. F49620-84-C-0115 for the U.S. Air Force Office of Scientific Research, Bolling Air Force Base. This final technical report covers the period September 1984 through September 1986, the duration of a Phase II SBIR award for 1984. The contract was monitored by Dr. A. Amos whose support we greatfully acknowledge.

The program manager at SEI for this project was Dr. Gilmer Blankenship and the principal investigator was Dr. William Bennett. Dr. H. G. Kwatny was co-investigator and technical advisor and Dr. N. Barkakati participated in the design of a computer software system for executive control of numerical and symbolic computations. Todd Aven and Ajay Virdy provided computer programming support for computations.

As part of this project five technical papers and conference presentations were made. These include references [19], [42], [75], [76], and

> W.H. Bennett, H.G. Kwatny, G.L. Blankenship, and N. Barkakati, "Continuum Modeling of Lattice Structures with Application to Vibration Control", *AIAA-86-0173*

which was presented at the AIAA 24th Aerospace Sciences Meeting, Jan. 6–9, 1986 in Reno, Nevada.

In addition, we are planning to submit two additional papers for presentation and publication. These are:

> W.H. Bennett and H.G. Kwatny "Continuum Modeling of Flexible Structures with Application to Vibration Control"

to be submitted to AIAA Journal, and

> W.H. Bennett, H.G. Kwatny, and T.S. Aven, "Practical Issues in Computation of Optimal Distributed State Feedback Control for Flexible Structures"

to be submitted to AIAA Guidance, Navigation, and Control Conference, August 1987, Monterey, California.

Contents

CONTENTS

List of Figures

List of Tables

Executive Summary

In this project we considered modeling the flexible dynamics of large space structures for the purpose of designing active control laws for the suppression of vibrations. We focus on distributed parameter models which capture the spatial dependency of the interaction between control forces, load forces (on board systems), exogenous distrubances, and sensors. Standard methods for such analysis used throughout the aerospace industry are based on finite-element analysis. In contrast in this project we focused on the use of continuum models for the elastic response of certain mechanical components like beams, cables, membranes. A composite model for a space structure may consist of several of these elastic elements interacting at their boundaries with rigid body elements. Effective continuum models for periodic truss structures may also be used to simplify the model construction. We have employed an asymptotic method of mutliple time and spatial scales called *homogenization* to compute such continuum models for a variety of truss structures under several assumptions.

Control for distributed systems centers on the definition of a distributed state for the mechanical system. A method for computation of a distributed feedback control gain is developed in detail including numerical algorithms and considerable testing is included.

In the first section we provide a technical introduction to the problem of control of distributed parameter systems arising from elastic mechanical structures. The second section contains a detailed discussion of a relatively new approach to computing distributed control gains for such systems. The method is based on a classical Wiener-Hopf problem and involves considerable numerical computations. We provide comparison with standard modal and finite-element methods and consider the practical limitation of the approach.

The third section contains a discussion of frequency response computations for continuum models arising from elastic dynamics modeled by partial differential equations. Composite structures are also considered in this section and it is shown that standard computations based on Laplace transforms can be used effectively to computed the required transform for rather complex interconnections between elastic and rigid components. In section four we consider homogenization of regular structures. The method is demonstrated on a simple one dimensional example which shows the existence of certain 'corrector' formulae which are important in computation of such models. A major question is the effectiveness of control laws developed for the resulting continuum model when applied to the finite structural lattice. In this section this question of combined homogenization and control is considered in some detail.

Finally section three contains several examples which illustrate the combined modeling and control methods proposed.

The problem of deriving complete transfer function representations of the structural models including transient response characterization by Green' functions is completely solved in this work. We show that the analysis of certain types structures—beams with one or more degrees of freedom—can be "automated" (though this is far from trivial) using a computer-based, symbolic manipulation system. The major question remains the

numerical evaluation of these functions. It is possible to use a variety of expansions and alternate algebraic representations for the functions for various ranges of their domains. This analysis can be supported with symbolic computation. Also, it appears likely that a truely functional methodology will require capability for rational approximation and interpolation of these functions. These issues limit the practical application of the method in question. Computer code for the symbolic and numerical algorithms are contained in the appendices.

1 Introduction

It is now generally accepted that large, low mass density lattice structures will be essential for several near term space applications. Moreover, it is apparent that active control of structural vibrations will be necessary to enhance their stiffness and damping properties. In this report we consider the construction of effective mathematical models for elastic dynamics of space structures with the objective of designing active control laws for these systems.

The success of active control for such structures will hindge to some extent on the ability of a control law to react to vibratory responses which may be initially localized before they propagate throughout a structure. This leads naturally to questions of how to implement active control so as to distribute the control effort spatially as it is needed. We argue that well known methods exist for the control of distributed parameter systems and can be effectively applied if continuum models for the candidate space structures can be computed. The nature of the required models is however quite different from the more standard finite element models which are popular for large structural analysis problems throughout the aerospace industry.

We begin in this section with a review of continuum models for active structural control. We highlight the nature of abstract state space models for these systems. In this study we have employed one method for the computation of a distributed control law for continuum dynamics based on a numerical procedure for spectral factoriztion. This method requires the computation of certain representation for an underlying state-space model for the structure to be controlled. In a later section we discuss the theoretical basis for computing effective distributed parameter models for large truss structures with random lattice infastructure. The method which involves "homogenization" (an asymptotic analysis of multiple scales) leads to the well known Timeshenko model for beam dynamics. The analysis provides formulae for the effective beam parameters which are quite different than have been suggested by other averaging schemes [1,2,3].

Comprehensive models of flexible spacecraft dynamics will involve systems with fairly complex interconnections of lumped and distributed subsystems, and therefore, we intend to construct the overall models by first developing subsystem models and then combining them according to the required interconnection rules. These interconnections lead to basic questions of causality and well-posedness of certain standard models for beams. These questions are crucial to the computation of hybrid, state-space modeling of an integrated space platform.

Throughout this effort we have focused on the potential for automatic and computer-aided computation of the models by a combination of modern computer algebra [4] (symbolic manipulation) and numerical methods. In our efforts we have used the program SMP [5,6]. We will review the progress in using SMP for the computation of irrational transfer function models for hybrid problems in a later section.

1.1 Generic Models for Structural Dynamics

In this section, we consider a generic model for elastic dynamics of structures from the point of view of continuum modeling. We will summarize the construction of a state space model and introduce a typical control problem for vibration suppression. We highlight the modal approximations which are popular for these problems and proceed to demonstrate an effective alternate technique for model construction and control computation based on the semi-group property [8] of a state space model. Effectively, modeling and control law computation can proceed in the frequency domain, based on transfer function methods, permitting the direct computation of a resolvent operator. We focus on the class of structural control problems for which the question of control of propagation of wave-like disturbances is important. In this framework, we can present the semi-group theory by concrete computations of practical interest to structural and control system engineers.

A popular continuum model for a flexible structure [9] is described by a system of partial differential equations (PDE)

$$m(z)\frac{\partial^2 w(t,z)}{\partial t^2} + D_0\frac{\partial w(t,z)}{\partial t} + A_0 w(t,z) = F(t,z) \tag{1}$$

where $w(t,z)$ is an N-vector of displacements of a structure Ω with respect to some equilibrium for Ω is a bounded, open set in $\Re^N$ [1]. The (vector) $z \in \Omega$ is a coordinate in Ω. We assume the boundary $\partial\Omega$ is smooth. The mass density $m(z)$ is positive definite and bounded on $\partial\Omega$. The damping term $D_0\partial w/\partial t$ models both (asymmetric) gyroscopic and (symmetric) structural damping effects. The internal restoring force $A_0 w$ is generated by a time-invariant, differential operator A_0 for the structure. For most common structural models, A_0 is an unbounded, differential operator with domain $D(A_0)$ consisting of certain smooth functions satisfying appropriate boundary conditions on $\partial\Omega$. Thus, for these problems, $D(A_0)$ is typically dense in the Hilbert space $\mathcal{H}_0 = L^2(\Omega)$ endowed with its natural inner product $\langle x,y\rangle_0 = \int_\Omega x^T(z)y(z)\,dz$. Often (but not always), the spectrum of A_0, $\sigma(A_0)$, consists of discrete eigenvalues with associated eigenfunctions which constitute a basis for $\mathcal{L}_2(\Omega)^2$.

The applied force distribution $F(t,z)$ can be thought of as consisting of three components

$$F(t,z) = F_d(t,z) + F_c(t,z) + F_a(t,z) \tag{2}$$

where F_d is N-vector of exogenous disturbances (possibly forces and torques), F_c is a continuous, distributed controlled force field (an available option in only some special applications), and F_a represents controlled forces due to localized actuation;

$$F_a(t,z) = \sum_{j=1}^{k} b_j(z)u_j(t) = B_0 u(t). \tag{3}$$

The actuator influence functions $b_j(z)$ are highly localized in Ω and can be approximated by Dirac delta functions. We assume that a finite number p of measurements can be made

[1] $\Re$ denotes the real numbers.

[2] $\mathcal{L}_2$ is the space of functions such that $f \in \mathcal{L}_p(\Omega)$ then $(\int_\Omega |f(\omega)|^p d\omega)^{1/2}$

as:

$$y(t) = C_0 w + C_0' \frac{\partial w}{\partial t} \tag{4}$$

where $y(t)$ is a p-vector. The operators $B_0 : \Re^m \rightarrow \mathcal{H}_0$, $C_0 : \mathcal{H}_0 \rightarrow \Re^p$, and $C_0' : \mathcal{H}_0 \rightarrow \Re^p$ are bounded.

The standard vibration control problem for this model is to find the controls $u_j(t)$, $j = 1, \ldots, k$ (we ignore the possibility of F_c) given the observations $y(t)$ to maintain the system state, e.g.,

$$x(t,z) = \begin{pmatrix} w(t,z) \\ \dot{w}(t,z) \end{pmatrix} \tag{5}$$

as close to its equilibrium state as possible.

1.1.1 State Space Models

The choice of state space given by (5) is often attractive for models in the generic form (1). (We will discuss later that attractive alternate state space models can arise in hybrid constructions.) A natural assumption for structural problems is that A_0 is symmetric with compact resolvent and discrete (real) spectrum which is bounded from below [9]. The state (5) can then be considered as an element of a Hilbert space $\mathcal{H} = D(A_0^{\frac{1}{2}}) \times \mathcal{H}_0$ with the energy norm

$$\|x\|_E^2 = \langle w, A_0 w \rangle_0 + \langle m\dot{w}, \dot{w} \rangle_0 \tag{6}$$

where the first term represents potential energy and the second term is kinetic energy. Thus an (abstract) *state space model* can be written

$$\begin{aligned} \dot{x}(t,z) &= Ax(t,z) + Bu(t) \\ y(t) &= Cx(t,z) \end{aligned} \tag{7}$$

where

$$A = \begin{bmatrix} 0 & I \\ -A_0 & -D_0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ B_0 \end{bmatrix}, \quad C = [C_0, C_0']. \tag{8}$$

For the elastic dynamics of space structures, there is always some (possibly small) damping D_0 appearing in (1) which causes A to be dissipative. Thus the criteria of the Hille-Yoshida-Phillips theorem [7] are satisfied and A generates a C_0-semigroup with an operator which we write suggestively as e^{At}. Moreover, such models are 'hyperbolic' [9] in the sense that the semigroup is a contraction, i.e., $\|e^{At}\| \leq 1$ and all but the zero frequency poles are only slightly damped; $\|e^{At}\| \leq e^{-\delta t}$ for some small $\delta > 0$. We remark that some popular models for structural elements such as beams with material damping may not fit in this framework [10,82]. However, this framework includes models appropriate for considerations of wave-like dynamics which propagate causally in the spatial domain. For such models, the question of how to compute controls $u(t)$ and system response $x(t)$ focuses on the so-called *mild solution* of (7);

$$x(t,z) = e^{At}x(0,z) + \int_0^t e^{A(t-\sigma)}Bu(\sigma)d\sigma. \tag{9}$$

For transient disturbances, the standard vibration control problem is essentially an optimal regulator problem which is readily solved by a linear state feedback which minimizes the performance index (parametrized by a real scalor $\epsilon > 0$)

$$J(u) = \int_0^\infty (\|y\|^2 + \epsilon\|u\|^2)dt \tag{10}$$

where $\|\cdot\|$ is the Euclidean norm on the appropriate finite dimensional space. This is the generic control problem surveyed in Balas [11]. In this paper, we will concentrate on the construction of state space models and computational aspects of equations of the form (1) and of optimal (discrete) controls $u_j(t)$ appearing in (3).

Various methods are available for approximation of the system (7) [12,13]. One popular method is based on a modal (eigen-) expansion of A which generates a sequence of finite dimensional subspaces $\mathcal{H}_k \subset \mathcal{H}, k = 1,2,\ldots$, where $\mathcal{H}_k = \text{span}\{\phi_j, j = 1,\ldots,k\}$ and the $\phi_j(z)$ are eigenfunctions (or mode shapes) for A [10,14]. Based on this approximation, a sequence of finite dimensional models for (5) can be generated;

$$\dot{x}^{(k)}(t) = A^{(k)}x^{(k)}(t) + B^{(k)}u(t). \tag{11}$$

Using a truncated model (11) with k finite and the performance index $J(u)$ projected onto the (finite dimensional) space $\mathcal{H}_k$, one can solve the associated optimal control problem for the first k modes of A. This is a classical approach and encompasses Ritz-Galerkin methods [10] as well as spline methods [12,13]. However, as noted in Balas [11] in all but a few special cases, the control law when applied to the system (5) will excite higher order modes. The inherent robustness and stability properties as well as the degree of suboptimality of control laws based on such truncated modal approximations has received a great deal of attention in both the engineering and mathematics literature [9,11]. Various alternate approaches are available which deal directly with infinite dimensional control problem given by (10) and (9)—at least abstractly (cf. Russell [15]). One method suggested by Davis [16,17] offers the advantage of a computational procedure for approximating the true optimal control in terms of the required control bandwidth. The method is based on an extension of a Weiner-Hopf solution [17] for the abstract control problem.

1.1.2 Wiener-Hopf Control

In the context of the regulator control problem given by the minimization of (10) subject to the infinite dimensional model (7), spectral factorization can be seen to provide an alternate solution [18] to a Riccati operator equation. In a series of papers, J. Davis and his students [16,17] have explored the application of spectral factorization for control design of a class of distributed parameter models for long trains with multiple locomotives. Also in Avramovic, et al. [19] considerations for application of these results to flexible structures were given. The details of this method are discussed in Section 2; however, in this section, we will summarize the relevant results and highlight the significance for modeling of flexible structures.

Taking Laplace transforms in (7) allows one to write (at least formally)

$$\hat{Y}(s,z) = CR(s;A)x(0,z) + CR(s;A)B\hat{U}(s). \tag{12}$$

The transfer function is $G(s) = CR(s;A)B$ where $R(s;A)$ is the resolvent operator for A, $R(s;A) : \mathcal{H} \to \rho(A) \subset \mathcal{H}$, where $\rho(A)$ is the resolvent set (or compliment to the spectrum of A).

The optimal control law which minimizes (10) subject to (7) consists of (linear) state feedback

$$u(t) = -B^*Kx(t,z) \tag{13}$$

where B^* is the formal adjoint of B in $\mathcal{H}$. We remark that B^*K is an (linear) integral operator on $\mathcal{H}$ which can be computed exactly without recourse to an infinite dimensional Riccati operator equation via the formula [17]

$$B^*K = \frac{1}{2\pi}\int_{-\infty}^{\infty}[F^*(i\omega)]^{-1}G^*(i\omega;A)CR(i\omega;A)d\omega, \tag{14}$$

where $F(s)$ is the unique, causal spectral factor of

$$I + G^T(-s)G(s) = F(s)F^T(-s). \tag{15}$$

For the infinite dimensional system (7), the transfer functions $G(s)$, $F(s)$ are irrational and $F(s)$ (resp. $F^T(-s)$) is analytic for $\mathcal{R}e\ s > 0$ (resp. $\mathcal{R}e\ s < 0$). Computational algorithms for (15) are given in Davis [17] and [42] where a numerical algorithm is given.

In this framework, the question of modeling of flexible structures for the design of feedback control for suppression of (linear) vibrations centers on computation of: (i) the irrational transfer function $G(s)$, and (ii) the resolvent $R(s;A)$. Then spectral factorization in (15) is performed using the Davis algorithm [17] and a numerical approximation to (14) can be obtained which is valid in the frequency bandwidth of the desired control action. We remark that although all computations are in the frequency domain, the resulting control law is a linear, distributed state feedback.

In this report we start in Section 2 with a detailed discussion of Wiener-Hopf method for computing distributed state feedback control. A numerical algorithm is presented and practical aspects of the method are considered. In Section 3 we provide a detailed discussion of frequency response calculations and modeling of hybrid structures consisting of distributed and lumped elements. In many cases of practical interest large repetitive truss structures can be effectively modeled as continua. Section 4 provides details of a method called *homogenization* and its application to this issue. New results are provided on the problem of control design based on the homogenized effective continuum model. Finally in Section 5 several examples of modeling and control computations are given. Appendices include computer code generated as part of this project and a short discussion of algorithms.

2 Wiener-Hopf Methods for Computation of Optimal, State-Feedback Control

The connections between least squares optimization, spectral factorization, and algebraic Riccati equations have been considered important in control theory for many years. (See e.g., Anderson [20], Brockett [18], Willems [21], Helton [22], and the references therein). To see how the connection arises, consider the standard, finite-dimensional, infinite time, quadratic regulator (LQR) problem:

$$\min_{u \in \mathcal{U}_{ad}} \int_0^\infty \|u(t)\|^2 + \|y(t)\|^2 dt \tag{16}$$

subject to the linear, time-invariant system model

$$\dot{x}(t) = Ax(t) + Bu(t),\ x(0) = x_0 \tag{17}$$

and controlled output

$$y(t) = Cx(t),\ t \geq 0. \tag{18}$$

The transfer function relating the Laplace transform of the input vector $\hat{u}(s)$ to the output vector $\hat{y}(s)$ is $G(s) = C[sI - A]^{-1}B$. The optimal control is known to be a linear state feedback $u(t) = -K_{opt}x(t) = -B^T P x(t)$ where P is the unique, positive definite symmetric solution to algebraic (matrix) Riccati equation,

$$PA + A^T P - PB^T BP + C^T C = 0. \tag{19}$$

Standard algebraic manipulations based on (16)–(19) provide the spectral factorization relation

$$\begin{aligned} [I \ + \ & K_{opt}(-sI - A)^{-1}B]^T [I + K_{opt}(sI - A)^{-1}B] \\ = \ & I + B^T(-s - A^T)^{-1}C^T C(sI - A)^{-1}B. \end{aligned} \tag{20}$$

[23, pp. 68] which we rewrite

$$H(s) = I + G^T(-s)G(s) = F^T(-s)F(s) \tag{21}$$

where $F(s) = I + K_{opt}[sI - A]^{-1}B$ is the *causal spectral factor* of $H(s)$.

An explicit integral equation can be derived for the optimal, state feedback under the additional assumption that the spectrum of the operator A in (17) is contained the open, left half-plane ($\mathbf{C}_-$).

Theorem 1 Given the optimal linear regulator problem as defined in (16)–(18) the optimal feedback gain (if it exists) can be computed as

$$K_{opt} = \frac{1}{2\pi} \int_{-\infty}^{\infty} [F^*(i\omega)]^{-1} G^*(i\omega) C R(i\omega, A) d\omega. \tag{22}$$

under the assumption that A is a stable (matrix) operator.

In (22) we use the notation $F^*(i\omega) = F^T(-i\omega)$ and $R(i\omega; A) = [i\omega I - A]^{-1}$ the (matrix) resolvent for A. One paradigm for computing LQR type control laws for distributed parameter systems is based on extending this computation (rather than the usual approach of finite element modeling followed by matrix computations). This requires the spectral factorization of $H(s)$ which may be rational even though the system is infinite dimensional. This will happen typically when "modal" control is the objective defined in (16); i.e., the operator C maps to a finite dimensional modal subspace (cf. Section 2.3 for details). The resolvent arising in (22) can be computed from a Green's function for the problem. In Section 3 we provide more details.

Proof of Theorem: From standard results [18] the optimal control (if it exists) will stabilize the closed loop system so that $\sigma(A - BK_{opt}) \subseteq \mathbf{C}_-$ where $\mathbf{C}_-$ is the open left half of complex plane. Construct a closed rectifiable contour Γ in the complex plane consisting of a relatively large portion of the $i\omega$-axis and a semicircular portion in the left half plane such that Γ encircles $\sigma(A - BK_{opt})$ in the positive sense. Let $A_c = A - BK_{opt}$. Then

$$\frac{1}{2\pi i}\oint_\Gamma [sI - A_c]^{-1}\, ds = I. \tag{23}$$

By assumption $\sigma(-A^T)$ is contained in $\mathbf{C}_+$ and

$$\frac{1}{2\pi i}\oint_\Gamma \left[sI + A^T\right]^{-1} ds = 0. \tag{24}$$

From (19) we write

$$A^T P + P A_c = -C^T C. \tag{25}$$

This leads to the relation

$$P(sI - A_c)^{-1} + (-sI - A^T)^{-1} P = (-sI - A^T)^{-1} C^T C (sI - A_c)^{-1}. \tag{26}$$

Now integrate (26) on Γ and use (23) and (24) to get

$$P = \frac{1}{2\pi i}\oint_\Gamma (-sI - A^T)^{-1} C^T C (sI - A_c)^{-1} ds. \tag{27}$$

Since the optimal gain is $K_{opt} = B^T P = [PB]^T$ we get

$$K_{opt} = \frac{1}{2\pi i}\oint_\Gamma B^T (sI - A_c)^{-1} C^T C (-sI - A^T)^{-1} ds. \tag{28}$$

Now from (20) and (21) we get that

$$C(sI - A_c)^{-1} B = C(sI - A)^{-1} B \left[I + K_{opt}(sI - A)^{-1} B\right]^{-1} \tag{29}$$

and therefore

$$C(sI - A_c)^{-1} B = G(s) F^{-1}(s).$$

Thus we can write

$$K_{opt} = \frac{1}{2\pi i}\oint_\Gamma F^{-T}(s) G^T(s) C (-sI - A)^{-1} ds. \tag{30}$$

Finally (22) is determined by substituting $s = i\omega$ under the observation that $G(i\omega) \to 0$ as $\omega \to \infty$.

□

For (22) to hold when $G(s)$ is irrational we must consider some limitation of the spectral properties of the associated infinite-dimensional operator A. Consideration for the computation of the Riccati operator P by the integral formula (22) suggests that the spectrum of A must consist of a countably infinite set of eigenvalues so that certain path integrals can be computed. The spectra $\sigma(A_c)$ and $\sigma(A)$ must be separated by the imaginary axis (including the point at infinity). Also, the observation that $G(i\omega) \to 0$ as $\omega \to \infty$ for rational functions must be replaced by a similar assumption which further restricts the class of irrational transfer functions [24]. As it turns out the usual assumptions used in constructing continuum models for mechanical structures serve to restrict the resulting transfer functions appropriately [51,27]. However, such transfer functions cannot be computed reliably from finite element models.

Effectively these additional assumptions will be guaranteed by the class of transfer functions for which spectral factorization can be performed. We will consider spectral factorization for distributed systems next. More importantly with the usual assumptions used in constructing continuum models for mechanical structures it appears that the resulting transfer functions will have the appropriate properties. However, in this study we have encountered much confusion in the literature. As discussed in the introduction a major goal of this study was to provide a consistent method for model construction leading to an appropriate class of transfer functions for models which are both well-posed and have appropriate spectral properties. In the next section we delineate the specific class of transfer functions for which spectral factorization can be computed efficiently by a numerical algorithm. We indicate the basis for the algorithm and discuss the implications of sampling and interpolation of the spectral factor. Finally we discuss the relationship between rational approximation and modal control.

2.1 Spectral Factorization by Frequency Sampling

In this section we review the basis for an interactive algorithm for computation of a frequency sampled spectral factor. The algorithm (due to Davis and Dickinson [17]) provides an effective computational tool for obtaining the optimal gain K_{opt} via (22) without regard to computational difficulties associated with large dimensional Riccati equations. Convergence of the algorithm depends on certain technical assumptions which delineate the class of transfer functions. In the finite dimensional setting, a recursive algorithm for computation of the causal spectral factor $F(s)$ follows from a Newton-Raphson iteration for the matrix Riccati equation (19) given an initial stabilizing feedback $K_0 = B^*P_0$;

$$P_{n+1}(A - BB^*P_n) + (A - BB^*P_n)^*P_{n+1} = -C^*C - P_nBB^*P_n.$$

At the n^{th} iteration one can take an approximation to the causal spectral factor as

$$F_n(s) = I + B^*P_n(sI - A)^{-1}B.$$

Then following Davis and Dickinson [17] this leads to the form of the algorithm

$$F_{n+1}(i\omega) = P_+ \left\{ [F_n^*(i\omega)]^{-1} H(i\omega) [F_n(i\omega)]^{-1} \right\} F_n(i\omega), \tag{31}$$

where P_+ is the causal projection operator defined on the convolution algebra $I \oplus \mathcal{L}_1$ or $I \oplus \mathcal{L}_2$ by

$$P_+ \left\{ I + \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \right\} = I + \int_0^{\infty} f(t) e^{-i\omega t} dt.$$

Computation of causal projection of a signal is a standard problem in signal processing which can be effectively solved through the use of a Hilbert transform [25], [26].

Definition 2 The Hilbert transform of a signal $f(t)$ is given as [25]

$$\check{f}(t) \equiv \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{f(\tau)}{t-\tau} d\tau = f(t) * \frac{1}{\pi t}. \tag{32}$$

The basic utility of the Hilbert transform is evident by examination of its Fourier transform.

Let $\check{F}(\omega)$ be the Fourier transform of $\check{f}(t)$ and $F(\omega)$ be the Fourier transform of $f(t)$.

$$\check{F}(\omega) = \int_{-\infty}^{\infty} \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{f(\tau)}{t-\tau} d\tau \, e^{i\omega t} dt. \tag{33}$$

Since for $Z(\omega) = \int_{-\infty}^{\infty} \frac{1}{\pi t} e^{i\omega t} dt$ we see that $H(\omega)$ is a complex function with the properties

$$\begin{aligned} |H(\omega)| &= 1 \\ \arg H(\omega) &= \begin{cases} -\pi/2, & \omega > 0 \\ \pi/2, & \omega < 0 \end{cases} . \end{aligned} \tag{34}$$

Therefore from (33) and (34) we get that

$$\check{F}(\omega) = \begin{cases} -iF(\omega), & \omega > 0 \\ iF(\omega), & \omega < 0 \end{cases} \tag{35}$$

Now to compute the causal projection of $f(t)$ (as in (31)) given frequency domain data $F(\omega)$ we first compute the Hilbert transform

$$\check{F}(\omega) = F(\omega) * \frac{1}{\pi\omega}.$$

By duality of the Fourier transform pair the property (35) holds in the time domain t;

$$\mathcal{F}^{-1} \left\{ \check{F}(\omega) \right\} = -i \, \mathrm{sgn}(t) f(t).$$

Finally causal projection can be computed as

$$P_+ \{F(\omega)\} = \frac{1}{2} \left[F(\omega) + i\check{F}(\omega) \right]. \tag{36}$$

Before we consider computational issues further we review the extension of this algorithm to irrational transfer functions. The questions of existence and uniqueness of the spectral factorization of the transform $H(s) = I + G^T(s)G(s)$ naturally lead to conditions for which $S(\omega) = G^T(-i\omega)G(i\omega)$ is positive semidefinite for ω real. In the convergence proof of the

iteration (31) Davis assumes that $G(s)$ is the Fourier transform of a real, vector-valued function which is both integrable and square integrable; i.e., $G(i\omega) \in \mathcal{F}(\mathcal{L}_1 \cap \mathcal{L}_2)$.

With these assumptions it is clear from the classical theory of Gohberg and Krein [28] that $H(s)$ has a unique spectral factorization as given in (21) with

$$F^{\pm}(i\omega) - I \in \mathcal{F}(\mathcal{L}_1^+)$$

where $\mathcal{F}(\mathcal{L}_1^+)$ is the class of Fourier transforms of functions in $\mathcal{L}_1$ with positive support. As noted in [17], the assumptions on $G(s)$ in fact imply that

$$F^{\pm 1}(i\omega) - I \in \mathcal{F}(\mathcal{L}_1^+ \cap \mathcal{L}_2^+)$$

and $F(i\omega) = F(-i\omega)$.

Using this assumption Davis is able to show that the recursion (31) starting from an initial $F_0(i\omega) \in \mathcal{F}(\mathcal{L}_1 \cap \mathcal{L}_2)$ has all iterates $F_n(i\omega) \in \mathcal{F}(\mathcal{L}_1 \cap \mathcal{L}_2)$ and that:

$$\lim_{n\to\infty} F_n^*(\omega)F_n(\omega) = H(\omega) \quad \text{almost everywhere} \tag{37}$$

$$\lim_{n\to\infty} F_n(s) = F(s) \qquad \text{for all } s \text{ with } \Re e\, s > 0. \tag{38}$$

These results should not be surprising for the class of transfer functions considered. The following theorems summarize well known properties of $\mathcal{L}_1$ and $\mathcal{L}_2$ functions.

Theorem 3 ([29]) If $f \in \mathcal{L}_1$ then

1. $\omega \mapsto \hat{f}(\omega)$ is uniformly continuous for $\omega \in \Re$
2. $\hat{f}(\omega) \in \mathcal{L}_\infty$
3. $|\hat{f}(\omega)| \to 0$ as $|\omega| \to \infty$
4. $f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \hat{f}(i\omega)e^{i\omega t}d\omega$ almost everywhere in $\Re$.

Theorem 4 (Parseval's theorem [29]) If $f \in \mathcal{L}_2$ then

1.
$$\int_{-\infty}^{\infty} |f|_2^2 dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} |\hat{f}(\omega)|d\omega$$

2. as $N \to \infty$
$$\int_{-N}^{+N} f(t)e^{-i\omega t}dt \xrightarrow{L_2} \hat{f}(i\omega)$$

3. as $N \to \infty$
$$\frac{1}{2\pi}\int_{-N}^{+N} \hat{f}(i\omega)e^{i\omega t}d\omega \xrightarrow{L_2} f(t)$$

So we see that $f \in \mathcal{L}_1$ means that $\hat{f}(i\omega)$ is bounded on ω and therefore the Fourier transform is well defined while $f \in \mathcal{L}_2$ provides consistent approximation theory for band limited signals. We note that the third $\mathcal{L}_1$ property means that such transfer functions are effectively band limited and strictly proper.

Application of spectral factor to the integral formula (22) will further restrict consideration to $G(s)$ both causal and stable so that $G(s)$ is analytic for $\Re e\ s > 0$. Thus the transfer functions we are considering belong to the Hardy space $G(s) \in \mathbf{H}^2 \cap \mathbf{H}^\infty$. Recall that by definition $\hat{f} \in \mathbf{H}^2$ if $\hat{f}$ is complex valued and analytic in $\mathbf{C}_+$ (the open right half of the complex plane) and

$$\sup_{\sigma>0} \int_{-\infty}^{\infty} |\hat{f}(\sigma + i\omega)| d\omega < \infty.$$

while $\hat{f} \in \mathbf{H}^\infty$ if $\hat{f}$ is complex valued and analytic in $\mathbf{C}_+$ and

$$\sup_{s \in \mathbf{C}_+} |\hat{f}(s)| < \infty.$$

The first property is inherited from $f \in \mathcal{L}_2$ while the second comes from $f \in \mathcal{L}_1$.

Finally, we remark that for mechanical structures that transfer function models can be factorized in a "product expansion" [30]

$$\hat{f}(s) = \prod_{k=1}^{\infty} \frac{1 + (s^2/z_k^2)}{1 + (s^2/\omega_k^2)}. \tag{39}$$

Thus we conclude these remarks by indicating that the class of transfer function models for which spectral factorization can be computed by the present method are meromorphic and have representations as (39).

2.1.1 Remarks on Algorithm Construction

With regard to the integral formula for the optimal state feedback gain it is clear that we need $[F(i\omega)]^{-1}$. Thus as suggested by Davis it is convenient to implement the iteration in the form

$$[F_{n+1}]^{-1} = [F_n]^{-1} \left(I + P_+ \left\{ [F_n^*]^{-1} H [F_n]^{-1} - I \right\} \right)^{-1}. \tag{40}$$

Furthermore by initializing with F_0 a diagonal matrix with diagonal elements equal to the spectral factors of the diagonal elements of H the second term of (40) remains a perturbation of the identity (since $[F_n^*]^{-1} H [F_n]^{-1} - I \to 0$) which regularizes the computations. The diagonal initialization guarantees that the first residual $[F_0]^{-1} H [F_0]^{-1}$ has ones on the diagonal and all off diagonal elements less than one in magnitude. Using the properties of the Hilbert transform and the formula (36) one can readily compute the causal spectral factor for the individual scalor transfer functions directly (i.e. without iteration). In particular, the k^{th} diagonal element of $H(\omega), h_k(\omega)$ is a real valued function with $h_k(\omega) > 0$. Let $\check{h}_k(\omega)$ be the Hilbert transform of $h_k(\omega)$; viz.,

$$\check{h}_k(\omega) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{h(\sigma)}{\omega - \sigma} d\sigma \tag{41}$$

then the causal spectral factor $h_k(i\omega) = f_k(-i\omega) f_k(i\omega)$ is given by

$$f_k(\omega) = \sqrt{\Re e\ h(\omega)}\ e^{-\frac{1}{2} \ln \check{h}(\omega)}. \tag{42}$$

Finally, we remark that numerical computation of the Hiblert transform can be achieved in several ways. Direct numerical integration of (41) is complicated by the fact that the integral is convergent in the Cauchy principal value sense. Effective quadrature algorithms for such problems have been coded and tested. A public domain version utilizing an adaptive quadrature procedure is available in the routine QAWC contained in a software package called QUADPACK [31].

Another approach is taken by Davis [17] who uses an algorithm of F. Stenger [32]. This procedure essentially implements a discrete (sampled) version of the required computation using a digital Hilbert transform. For a finite number of sample points the Hilbert transform computation can be implemented by taking a discrete "fast fourier transform" (FFT) of the sampled data and shifting the imaginary part of the transformed data according to (33). It is well known that control of error induced by the sampling process (Gibbs phenomenon) requires the careful choice of "data windows" or weighting functions for the computation. Although these considerations are well described in the literature on digital signal processing [26], [34], [33] it is not apparent that these considerations were contained in the work of F. Stenger.

In our experience the direct implementation of the algorithm described in Davis and Dickinson, based on the causal projection of F. Stenger [32] is unreliable at best. Since detailed design of window functions for discrete Hilbert transforms was considered outside the scope of the present study we have found it expedient to use the adaptive quadrature software from QUADPACK [31]. It should be noted however that this approach may suffer in production grade applications by the computational inefficiency of the quadrature procedure by comparison with an FFT implementation of the discrete Hilbert transform.

2.2 Importance of Damping in Models for Distributed Control

For the present study we have attempted to compute distributed controls for several examples. Several negative results lead us to reevaluate the theoretical basis for control and modeling problem. Our difficulties stem from precise computation of the irrational transfer functions for several distributed models which will be discussed later. Thus our control laws are computed and tested on bona fide distributed parameter models and not on finite dimensional approximations; therefore the idiosyncracies of many available models (which we unfortunately tried first) lead to difficulties. In this section we briefly review our most recent conclusions about the assumptions in modeling for mechanical structures and resulting computation of optimal control laws.

The most pertinent comments in the literature appear to be summarized in the work of Gibson [10]. In particular, the importance of damping and the ability to compute distributed control laws for infinite dimensional models for mechanical structures. It is true that computation of control laws for such problems will always involve approximation in modeling, control problem formulation, and numerical computation. Gibson focuses on this question of approximation in modeling and control for distributed parameter problems via the use of finite dimensional models and control computation based on these models. Although, on the surface, the Davis method (based on spectral factorization of irrational

transfer functions) is not subject to finite dimensional approximation (at least in the sense of modal expansions) it is clear that approximation via sampling and interpolation of the transfer functions involved is required. The examples in Section 5 suggest that additionally some form of rational approximation may also be required for numerical computation of the distributed gain.

Let us consider some observations of Gibson [10]. First, Consider the class of elastic mechanical structures modeled,

$$\ddot{x}(t,z) + C_0\dot{x}(t,z) + A_0x(t,z) = Bu(t) \tag{43}$$

where $x \in \mathcal{H}(\Omega)$ appropriately chosen to match required boundary conditions, $u(t)$ is a finite dimensional vector of controls, A_0 is a self adjoint operator $A_0 : D(A_0) \to \mathcal{H}$ densely defined on $\mathcal{H}$. Gibson further assumes that A_0 is *coercive*; i.e.

$$\langle A_0x, x\rangle_{\mathcal{H}} \geq \rho^2\|x\|^2_{\mathcal{H}}, \quad x \in D(A_0)$$

and A_0^{-1} is compact. C_0 is nonnegative, symmetric linear operator and there exists $\gamma \geq 0$ such that

$$\|C_0x\| \leq \gamma^2\|A_0x\|_{\mathcal{H}}, \quad x \in D(A_0). \tag{44}$$

Finally, B_0 is taken to be a bounded operator. Gibson shows that (44) is a necessary condition for the resulting semigroup operator for

$$A = \begin{bmatrix} 0 & I \\ -A_0 & -C_0 \end{bmatrix}$$

on $\mathcal{H} \times \mathcal{H}$ to be *uniformly exponentially stable*; i.e., there exists $M > 0, \alpha > 0$ such that $\|e^{At}\| \leq Me^{-\alpha t}$ for $t \geq 0$. Such behavior corresponds to the case when damping provides a uniform decay rate. Gibson addresses the question of convergence and stability of a sequence of finite dimensional (possibly modal) approximate control problems to the unique optimal control for the distributed model. His results indicate [10, theorem 4.1] that for the quadratic linear regulator problem for the distributed model above (43) that if a solution exists the exact optimal feedback control provides a closed loop system whose infinitesimal generator $A_{cl} = A + BK_{opt}$ generates a strongly continuous semigroup which is uniformly exponentially stable.

For systems without damping; i.e. $C_0 \equiv 0$ in (43), Gibson shows that there can be no nonnegative, selfadjoint solution for the algebraic Riccati equation for the distributed model. This follows from the observation [10, theorem 5.13] that for $C_0 \equiv 0$, the semigroup generated by $A + \mathcal{E}$ for any $\mathcal{E}$ a *compact* linear operator, cannot be uniformly exponentially stable.

For systems with damping Gibson shows that the damping must be such that A (the generator for the open loop system) is uniformly exponentially stable in order that uniform exponential stability of the closed-loop system can be obtained by compact linear feedback. The significance of this fact is that the physical nature of damping must be considered in the computation of distributed parameter control and for problems where the available control affectors provide 'localized' (in the spatial domain) forces (or torques). Damping

with (at least) a uniform decay rate (for essentially all modes) is required for such systems to be controlled with uniform exponential stability. Although it is generally agreed that physical structures will have this property it is somewhat disconcerting that many standard models for simple structural elements like beams with internal damping *do not* have these properties. In fact, with the exception of viscous damping there does not appear to be a single, universally accepted, well-defined mathematical model for beam dynamics which is obviously appropriate for the study of distributed parameter control of structures [82].

In Section 3 we will discuss damping models for structural elements in detail. At this point we remark that in this study it became apparent that the performance of active control of elastic structures was seen to be heavily dependent on the type of damping models used. If one is willing to assume a sufficient amount of viscous damping then stabilizing control can be computed. Since viscous effects will definitely not be present in space applications one is faced with a choice of several models for internal damping. Many of these models lead to problems for which either a uniform decay rate is not available or for which the spectrum of the operator contains more than mere eigenvalues. These models cannot be stabilized by compact linear feedback whether it is computed by Weiner-Hopf methods or by modal approximation and solution of a finite-dimensional quadratic regulator for the reduced model. However if one takes the second path the relevant stability questions are completely lost in the model reduction.

What is an appropriate choice for modeling internal damping in space structures is definitely an open question. This issue has apparently not received a great deal of attention in the aerospace industry especially in that portion of the community involved in control of large flexible structures. The standard procedure here (as evidenced in for instance the ACOSS program) is to assume "low" modal damping can be added to the reduced-order model prior to control system design. In the present frequency domain computations one is forced to resolve these issues before proceeding with control computation.

For the computation of distributed control according to (30) it is therefore necessary that damping be of (at least) uniform exponential type. However, even more is required in practice. Since K_{opt} will be computed (in our approach) by numerical evaluation of (30) it is clear that the integration will be performed over a finite bandwidth by numerical quadrature. The required bandwidth for integration depends crucially on the open loop system bandwidth for the continuum structural model. This means that effectively damping must be *ultimately proportional to frequency* for all high frequency modes. This is consistent with the usual notion of material damping but is required here for numerical reasons.

2.3 Practical Aspects of Computing Distributed Control Laws

In this section we review the computational aspects of Wiener-Hopf control based on spectral factorization and distributed gain computation and its relation to standard methods based on finite dimensional approximation of the evolution equations. We highlight various methods for model approximation (i.e. reduced order modeling). The methods we employ are based on frequency domain computations for state space models and can be

compared directly with state space methods. However, as discussed in Youla et al [35] the Wiener-hopf method can treat models which have no state space analog [24].

Reduced order modeling (ROM) of the abstract control problem (7)–(10) can proceed in many ways [36]. At a fundamental level analysis of ROM involves decomposition of the state space $\mathcal{H}(\Omega)$ via an orthogonal decomposition $\mathcal{H} = \mathcal{H}_N \oplus \mathcal{H}_R$ where $\mathcal{H}_N$ is a finite dimensional subspace of $\mathcal{H}(\Omega)$ and $\mathcal{H}_R$ is a *residual space*. For example, with $x(z) \in \mathcal{H}(\Omega)$ let x be expanded as

$$x(z) = \sum_{k=1}^{\infty} \tilde{x}_k \phi_k(z)$$

which we assume converges. The series $\{\phi_k(z)\}_{k=1}^{\infty}$ provides an orthogonal basis for $\mathcal{H}(\Omega)$. Then the ROM space is

$$\mathcal{H}_N(\Omega) = \text{span}\{\phi_1(z), \phi_2(z), \ldots, \phi_N(z)\}.$$

This approach embodies finite element methods based on Ritz-Galerkin approximation as well as truncated modal approximation.

In Balas [36] tradeoffs are discussed for the formulation of optimal control for distributed parameter systems (DPS) by projection onto reduced-order subspaces. In Bernstein and Hyland [37] an optimal projection approach is described for reduced-order models. In applications it may be desirable to define a control objective of the form (10) so that only a finite number of modes are considered for active control. Typically the accuracy of any mathematical model degrades with higher frequencies. Sensors and actuators have finite bandwidth.

Let P_N be an orthogonal projection $P_N : \mathcal{H}(\Omega) \rightarrow \mathcal{H}_N(\Omega)$. The quadratic control objective (10) is defined with respect to a *controlled output* $y(t) = Cx(t,z)$ given by the operator $C : \mathcal{H} \rightarrow \Re^p$. A *reduced-order control objective* can be defined by the choice $C = P_N$. For the finite dimensional version of the control problem (7)–(10) it is known that an optimal control law exists which stabilizes the system if and only if (A, B) is *stabilizable* (i.e., any state trajectory contained in the uncontrollable subspace is exponentially stable) and (A, C) is *detectable* (i.e., any state trajectory contained in the unobservable subspace is exponentially stable.) For distributed parameter models for elastic structures we assume the operator A has discrete spectrum and is at least, uniformly exponentially stable for essentially *all* modes. Thus the optimal control problem (7)–(10) with controlled output given by $C = P_N$ has a unique stablizing control law given by (13).

We remark that the formulation is technically distinct from the methods discussed in [36] in that the control problem incorporates ROM as part of the control objective rather by approximation (ROM) of the (constraint) equation (7) by Ritz-Galerkin or modal truncation. In terms of the mild solution (9) we see that with $y(t) = P_N\, x(t,z)$,

$$y(t) = P_N e^{At} x(0,z) + \int_0^t P_N e^{A(t-\sigma)} B u(\sigma) d\sigma. \tag{45}$$

Thus the control problem can be equivalently stated:

Problem minimize the objective

$$J(u) = \int_0^{\infty} \|y\|^2 + \|u\|^2 \, dt$$

subject to (45).

The projection P_N is orthogonal so that for any $x(z) \in \mathcal{H}(\Omega)$ we can write $P_N x + P_R x$ where P_R is the orthogonal complement projection; i.e.,

$$P_R x(t,z) = \sum_{k=N+1}^{\infty} \tilde{x}_k(t)\phi_k(z).$$

Thus (45) can be written alternately as an evolution

$$\dot{y}(t) = P_N A P_N x(t,z) + P_N A P_R x(t,z) + P_N B u(t).$$

Roughly put, the term $P_N A P_R x(t,z)$ does not contribute to the control objective so that the control problem is equivalent to the problem

$$\min_{u \in U_{ad}} \int_0^{\infty} \|u(t)\|^2 + \|y(t)\|^2 dt$$

subject to

$$\dot{y}(t) = P_N A y(t) + P_N B u(t);$$

i.e., a finite-dimensional control problem.

The Wiener-Hopf approach, taken in this study, with the reduced-order control objective further clarifies that the underlying control problem is finite-dimensional. One way to view ROM in the frequency domain is to consider rational approximation of the transcendental functions. Consider the transform of (9);

$$\hat{x}(s,z) = R(s;A)x^0(z) + R(s;A)B\,\hat{u}(s).$$

In Section 3 we show that for elastic structures this abstract equation can be written in the form

$$\hat{x}(s,z) = \int_{\Omega} G_r(s;z,w)x^0(w)dw + H_{BC}(s,z)\hat{u}(s),$$

where $G_r(s;z,w)$ is a *Green's function*. The resolvent operator $R(s;A)$ is an integral operator with kernel a Green's function. Let ω_k, ζ_k be the k^{th} modal frequency and damping for A with respect to some ROM basis for $\mathcal{H}_N(\Omega)$. Then (at least on the surface) modal truncation can be viewed as a truncated partial fraction expansion (see Wie and Bryson [30]);

$$P_N\,\hat{x}(s,z) = \int_{\Omega} G_{r,N}(s;z,w)x^0(w)dw + H_{BC,N}(s;z,w)\hat{u}(s)$$

where

$$G_{r,N}(s;z,w) = \sum_{k=1}^{N} \frac{\psi_k(z,w)}{s^2 + 2\zeta_k\omega_k s + \omega_k^2}$$

$$H_{BC,N}(s;z) = \sum_{k=1}^{N} \frac{\xi_k(z,w)}{s^2 + 2\zeta_k\omega_k s + \omega_k^2}.$$

The computational procedure for the Wiener-Hopf problem outlined in Section 2 starts with the definition of the transfer function $G(s)$ from the control $\hat{u}(s)$ to the controlled output $\hat{y}(s)$. For the case of ROM control objective we get

$$\begin{aligned} G(s) &= P_N R(s;A)B \\ &= H_{BC,N}(s;z) \end{aligned}$$

a *rational* transfer function in the complex frequency s. Several algorithms exist for this classical problem [38,39]. Then the optimal gain formula requires integration as

$$\begin{aligned} K_{opt} &= \frac{1}{2\pi}\int_{-\infty}^{\infty}[F^*(i\omega)]^{-1}G^*(i\omega)CR(i\omega;A)d\omega \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}[F^*(i\omega)]^{-1}G^*(i\omega)C\left(\int_\Omega G_r(i\omega;z,w)\cdot dw\right)d\omega. \end{aligned} \tag{46}$$

We remark that the feedback "gain" is an integral operator with kernel we call the *distributed gain* K_{dis} where

$$K_{dis}(w) = \frac{1}{2\pi}\int_{-\infty}^{\infty}[F^*(i\omega)]^{-1}G^*(i\omega)CG_r(i\omega;z,w)\,d\omega \tag{47}$$

can be computed as a path integral in the complex plane and replacing the Green's function G_r with its ROM $G_{r,N}$ and G replaced with G_N in (47) shows that the integrand is a rational function in s. Thus $K_{dis}(w)$ can be evaluated by residues.

The main point to be emphasized is that the numerical procedure discussed in Section 2 takes a fundamentally different approach by using frequency sampling *in place of* rational approximation for the required computation of:

- spectral factorization
- distributed gain computation

The numerical procedure can provide an effective, computationally efficient alternative—even for large finite-dimensional problems which arise in optimal control of DPS with objective which weight a ROM subspace.

Finally, we conclude this section with some remarks about the practical aspects of frequency sampling and numerical computations. The spectral factorization algorithm has already been discussed in detail. The numerical evaluation of $K_{dis}(w)$ proceeds by a trapezoidal quadrature over a finite bandwidth $-\omega_0 \le \omega \le \omega_0$. The choice of ω_0 for numerical integration depends on the bandwidth of the transfer function $G(s)$. For the case of reduced-order control objective ω_0 can be readily determined by reference to the rational transfer function. For the irrational $G(s)$ it is necessary that the high frequency modes be sufficiently damped so that finite bandwidth integration yields a reasonable approximation to the true $K_{dis}(w)$. The examples in Section 5 serve to illustrate that in some cases even more seems to be required for effective convergence of numerical algorithms. Depending on the choice of the control objective it seems that rational approximation may still be required for the finite bandwidth computation of the distributed gains.

The frequency domain approach offers several alternatives for model order reduction, approximation, and control computation. Direct approximation of transfer functions by truncation of product expansions is discussed in Wie and Bryson [30]. This method has also been used for simulation [40]. Here we expand an irrational transfer function (assumed to be meromorphic)

$$G(s) = \prod_{k=1}^{\infty} \frac{n_k(s)}{d_k(s)} \tag{48}$$

where $n_k(s), d_k(s)$ are (relatively low order) polynomials in s. By truncation of (48) we effectively match a finite number of both poles and zeros of the irrational transfer function. This method is clearly distinct from the modal and finite element methods. Approximation of the zeros of the transfer function may be extremely important for control system design since performance of designs based on poorly approximated zero locations can be very sensitive to model errors [30,24]. Zero locations arise in flexible structure control problems from the location and type of actuators and sensors.

Finally, in this report we have not addressed the design of *dynamic* control laws. This question is discussed in detail in [24]. We remark that distributed state feedback control may be quite realistic for the next generation of spacecraft with flexible structures. Recent technology developments in instrumentation for distributed measurement clearly shows a trend to provide low cost alternatives for distributed measurements of displacements and angular rates as well as various physical quantities such as strain rates [41]. In Section 3 we consider choice of state space coordinates for continuum models of structures including the physical significance of the state variables.

3 State Space Models for Distributed Structural Elements

3.1 Standard Forms for Linear PDEs

In this section, we will consider the problem of deriving standard state-space descriptions for typical distributed elements. In Section 3.2 we use these descriptions to compute the complete, frequency domain response as required for the control problem outlined above. It is our contention that the systems of interest to us—specifically beams with one space variable and perhaps several degrees of freedom—can be represented by one of two standard forms. Once identifying the structure of these standard models, it is straightforward, although far from trivial, to mechanize the construction of the required transfer matrices using symbolic computation [42]. Moreover, in order to assemble hybrid system models by the interconnection of components or subsystems, it is essential to have a clear understanding of the causal requirements of the component mathematical models. The following paragraphs develop the required concepts in terms of commonly used structural elements. Since typical elements interact at physical boundaries, our foremost concern is with the formulation of appropriate boundary conditions for well-posed, initial-boundary value problems.

Before proceeding, we establish some basic terminology associated with systems of partial differential equations. Consider the system of first-order, partial differential equations defined for $t \geq 0$ and $0 \leq z \leq L$

$$E\frac{\partial w}{\partial t} = \tilde{F}\frac{\partial w}{\partial z} + \tilde{H}w, \quad w \in R^n. \tag{49}$$

If E is nonsingular, then (49) can be written

$$\frac{\partial w}{\partial t} = F\frac{\partial w}{\partial z} + Hw \tag{50}$$

where $F = E^{-1}\tilde{F}$, $H = E^{-1}\tilde{H}$. If F has only real eigenvalues and a complete set of eigenvectors, then the system is said to be *hyperbolic* (see, for example, Zauderer [43]). If there are multiple real eigenvalues and less than a complete set of eigenvectors, then the system is of (partial) *parabolic* type. If all of the eigenvalues are complex, the system is of *elliptic* type. Systems with complex eigenvalues are not causal. Lyczkowski, et al. [44], and Sursock [45] provide an interesting discussion of this point in connection with a fluid flow problem. The underlying problem is that systems with complex eigenvalues are not well-posed as initial value problems, John [46], Lax [47]. We will not consider such problems any further.

If E is singular, (49) can give rise to mixed systems of all types. Some examples can be found in Firedly [48] and Lapidus and Pinder [49]. Our interest in this case will be limited to purely parabolic systems of the type

$$\frac{\partial w}{\partial t} = G\frac{\partial^2 w}{\partial z^2} + F\frac{\partial w}{\partial z} + Hw \tag{51}$$

which commonly arise in engineering problems.

When the equations of motion for structural elements are derived from conservation laws—in particular, from variational principles—the resulting equations are typically of hyperbolic type (see, for example, Crandall, et al. [27]). However, further standard assumptions and approximations reduce the equations to parabolic systems in the form of (51). In the following paragraphs, several examples will be given. In summary, we are primarily interested in hyperbolic systems (49) and parabolic systems (51). In addition to equations (49) or (51), there are associated initial and boundary conditions. For equation (49), these conditions take the general form

$$\begin{array}{ll} \text{initial conditions} & w(z,0) = f(z) \\ \text{boundary conditions} & \Sigma_1 w(0,t) + \Gamma_1 w(L,t) = g(t) \end{array} \tag{52}$$

where $\dim(g)=\dim(w)$; and for equation (51), they take the general form

$$\begin{array}{ll} \text{initial conditions} & w(z,0) = f(z) \\ \text{boundary conditions} & \Sigma_1 w(0,t) + \Sigma_2 \frac{\partial w}{\partial z}(0,t) + \Gamma_1 w(L,t) + \Gamma_2 \frac{\partial w}{\partial z}(L,t) = g(t) \end{array} \tag{53}$$

where $\dim(g)=2\dim(w)$.

It is well known that the coefficient matrices in (52), (53) must satisfy certain constraints if the problem formulation is to be well-posed. In the hyperbolic case (equations (49) and (52)), these constraints essentially require that the boundary conditions be compatible with the wave directions. Further discussion can be found in Russell [15] and Agarwala [13]. In the following sections we will discuss some standard models for beams. The following notation is standard and assumes the elastic beam is uniform (i.e., the parameters are independent of the spatial coordinate.)

Standard Notation for Uniform Beam Formulae

κG	effective shear modulus
ρ	mass density
A	cross sectional area
E	modulus of elasticity
I	moment of inertia
L	beam length
z	longitudinal coordinate
$\eta(z)$	lateral displacement
ϕ	angular rotation of cross section

3.1.1 The Timoshenko Beam Model

We will show how some conventional beam models can be reduced to the standard forms described in the preceding paragraphs. In particular, we will begin with the Timoshenko model and then consider two commonly used approximations which can be derived from it, the Euler-Bernoulli model and the 'string' model. The equations of motion can be derived

using Lagrange's equations [27] and in the absence of dissipation take the form

$$\begin{aligned} \rho A \frac{\partial^2 \eta}{\partial t^2} &= \frac{\partial}{\partial z}\left[\kappa G A\left(\frac{\partial \eta}{\partial z} - \phi\right)\right] \\ \rho I \frac{\partial^2 \phi}{\partial t^2} &= \frac{\partial}{\partial z}\left[EI\frac{\partial \phi}{\partial z}\right] + \kappa G A\left(\frac{\partial \eta}{\partial z} - \phi\right) \end{aligned} \tag{54}$$

along with the natural boundary conditions for $\alpha = 0, L$

$$\begin{array}{cc} \text{displacement} & \text{or} \quad \text{shear force} \\ \eta(\alpha,t) = \eta_\alpha(t) & \kappa G A\left(\frac{\partial \eta(\alpha,t)}{\partial z} - \phi(\alpha,t)\right) = f_\alpha(t) \end{array} \tag{55}$$

and

$$\begin{array}{cc} \text{rotation} & \text{or} \quad \text{moment} \\ \phi(\alpha,t) = \phi_\alpha(t) & EI\left(\frac{\partial \phi(\alpha,t)}{\partial z}\right) = \tau_\alpha(t). \end{array} \tag{56}$$

The two equations (54) can be replaced by four, first-order equations by introducing two new variables, $\nu(z,t)$ and $\gamma(z,t)$:

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= \frac{\partial \nu}{\partial z}, \\ \frac{\partial \nu}{\partial t} &= \frac{\kappa G}{\rho}\left(\frac{\partial \eta}{\partial z} - \phi\right), \\ \frac{\partial \phi}{\partial t} &= \frac{\partial \gamma}{\partial z} + \frac{A}{I}\nu, \\ \frac{\partial \gamma}{\partial t} &= \frac{E}{\rho}\frac{\partial \phi}{\partial z}. \end{aligned} \tag{57}$$

These equations clearly represent a hyperbolic system and the natural boundary conditions become for $\alpha = O, L$

$$\begin{array}{cc} \text{displacement} & \text{or} \quad \text{shear force} \\ \eta(\alpha,t) = \eta_\alpha(t) & \nu(\alpha,t) = \nu_\alpha(t), \dot{\nu}_\alpha(t) = \frac{f_\alpha(t)}{\rho A} \end{array} \tag{58}$$

and

$$\begin{array}{cc} \text{rotation} & \text{or} \quad \text{moment} \\ \phi(\alpha,t) = \phi_\alpha(t), & \gamma(\alpha,t) = \gamma_\alpha(t),\ \dot{\gamma}_\alpha(t) = \frac{\tau_\alpha(t)}{\rho I}. \end{array} \tag{59}$$

Note that the boundary conditions applied to the first order system (57) require the time integral of boundary forces or moments applied to the beam. It is easy to confirm that the transfer functions relating forces or moments to displacements or rotations as derived from either equations (54) or (57) are indeed identical and that the required integration of the shear force or moment is essential in the first-order forms.

3.1.2 The Bernoulli-Euler Beam Model

The Bernoulli-Euler model is obtained from the Timoshenko model with two additional assumptions:

1. rotational inertia is neglected, $\rho I \to 0$
2. shear deformation is neglected, $\frac{\partial \eta}{\partial z} - \phi \to 0$

Assumption 1. reduces the second of equations (54) to

$$\kappa GA\left(\frac{\partial \eta}{\partial z} - \phi\right) = -\frac{\partial}{\partial z}\left(EI\frac{\partial \phi}{\partial z}\right). \tag{60}$$

Equation (60) and assumption 2. are now used to reduce the first equation of (54) to

$$\rho A\frac{\partial^2 \eta}{\partial t^2} = -\frac{\partial^2}{\partial z^2}\left(EI\frac{\partial^2 \eta}{\partial z^2}\right). \tag{61}$$

Note that equation (60) along with assumption 2 leads to the following expression for shear force

$$f = \kappa GA\left(\frac{\partial \eta}{\partial z} - \phi\right) = -\frac{\partial}{\partial z}\left(EI\frac{\partial^2 \eta}{\partial z^2}\right). \tag{62}$$

Although (62) is commonly used in conjunction with the Bernoulli-Euler model (61), it should only be used with caution. Equation (61) is valid only in the limit $f \to 0$. We will return to this point below.

The boundary conditions (55) and (56) reduce to for $\alpha = 0, L$

$$\begin{array}{cccc} \text{displacement} & \text{or} & \text{shear force} \\ \eta(\alpha,t) = \eta_\alpha(t), & & -EI\frac{\partial^3 \eta(\alpha,t)}{\partial z^3} = f_\alpha(t), \end{array} \tag{63}$$

and

$$\begin{array}{cccc} \text{displacement} & \text{or} & \text{moment} \\ \frac{\partial \eta(\alpha,t)}{\partial z} = \phi(\alpha,t) = \phi_\alpha(t), & & EI\frac{\partial^2 \eta(\alpha,t)}{\partial z^2} = \tau_\alpha(t) \end{array} \tag{64}$$

Note that nonzero shear force is included as an admissible boundary condition; however, the remarks following equation (62) apply.

Equation (61) can be reduced to 'first-order' form by introducing a new variable $\gamma(z,t)$

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= -\frac{I}{A}\frac{\partial^2 \gamma}{\partial z^2}, \\ \frac{\partial \gamma}{\partial t} &= \frac{E}{\rho}\frac{\partial^2 \eta}{\partial z^2}, \end{aligned} \tag{65}$$

and the boundary conditions associated with (65) are for $\alpha = 0, L$:

$$\begin{array}{cccc} \text{displacement} & \text{or} & \text{shear force} \\ \eta(\alpha,t) = \eta_\alpha(t), & & \frac{\partial \gamma(\alpha,t)}{\partial z} = \gamma'_\alpha(t), \dot{\gamma}'_\alpha = -\frac{A}{I} f_\alpha(t), \end{array} \tag{66}$$

and

$$\begin{array}{cccc} \text{rotation} & \text{or} & \text{moment} \\ \frac{\partial \eta(\alpha,t)}{\partial z} = \phi_\alpha(t), & & \gamma(\alpha,t) = \gamma_\alpha(t),\ \dot{\gamma}_\alpha = \frac{\tau_\alpha(t)}{\rho I}. \end{array} \tag{67}$$

Observe that (65) is a parabolic system of the type (51). Equations (65)–(67) can be derived directly from (61) or from (57) upon invoking assumptions 1 and 2. We should also note that a corresponding expression for shear force becomes

$$f = -\left(\frac{I}{A}\right)\frac{\partial^2 \gamma}{\partial t \partial z}. \tag{68}$$

3.1.3 The 'String' Model

In some situations, bending deformation may be negligible with respect to shear deformation, that is, $|\phi| \ll |\partial\eta/\partial z|$. In this case, the first equation of (54) reduces to

$$\rho A \frac{\partial^2 \eta}{\partial t^2} = \frac{\partial}{\partial z}\left(\kappa G A \frac{\partial \eta}{\partial z}\right) \tag{69}$$

with boundary conditions for $\alpha = 0, L$

$$\begin{array}{ccc} \text{displacement} & \text{or} & \text{shear force} \\ \eta(\alpha,t) = \eta_\alpha(t) & & \kappa G A \frac{\partial \eta(\alpha,t)}{\partial z} = f_\alpha(t). \end{array} \tag{70}$$

This simple model is primarily useful for illustrative purposes. Again by introducing the new variable $\nu(z,t)$, equation (69) can be replaced by two, first-order equations

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= \frac{\partial \nu}{\partial z} \\ \frac{\partial \nu}{\partial t} &= \frac{\kappa G}{\rho}\frac{\partial \eta}{\partial z} \end{aligned} \tag{71}$$

subject to boundary conditions for $\alpha = 0, L$

$$\begin{array}{ccc} \text{displacement} & \text{or} & \text{shear force} \\ \eta(\alpha,t) = \eta_\alpha, & & \nu(\alpha,t) = \nu_\alpha(t),\ \dot{\nu}_\alpha = \frac{f_\alpha(t)}{\rho A}. \end{array} \tag{72}$$

3.2 Beam Models with Dissipation

Various dissipation models have been proposed for use with the Timoshenko and Bernoulli-Euler beam models. A summary of the most frequently cited models may be found in Pichè [53] and Wie and Bryson [30]. Experimental data is scant, particularly in the high frequency range, so that none of these models can be considered definitive. Recent experimental work by Russell has demonstrated that the usual conjecture of 'damping proportional to frequency' for internal (material) dissipation is *not* complete [82]. However, it is reasonably representative of material damping for certain low to midrange frequencies. Moreover, some of the more popular models were never intended for use in general transient analysis and fail to yield well-posed dynamical models (Chen and Russell [54]). One approach to developing dynamical models which include dissipation is to augment the variational development of the equations of motion by introducing a Rayleigh dissipation function (see also [82]. We formulate such a function based on the following assumptions:

1. external dissipation forces are proportional to the coordinate velocities (i.e. $\dot{\eta}, \dot{\phi}$),
2. internal dissipation forces are proportional to strain rates (i.e., shear strain rate, $\dot{\gamma} = \partial\dot{\eta}/\partial z - \dot{\phi}$, and compressive strain rate $\partial\dot{\phi}/\partial z$).

$c_4 \frac{\partial^4}{\partial z^4}\dot{\eta}$	Kelvin-Voight damping
$-c_2 \frac{\partial^2}{\partial z^2}\dot{\eta}$	Chen-Russell (or 'square root') damping
$c_1\dot{\eta}$	viscous damping

Table 1: Dissipation terms and standard damping mechanisms

Thus, we define the dissipation function as

$$R(\dot{\eta},\dot{\phi}) \equiv \frac{1}{2}\int_0^L \left[c_1\left(\frac{\partial \eta}{\partial t}\right)^2 + c_2\left(\frac{\partial \phi}{\partial t}\right)^2 + c_3\left\{\frac{\partial}{\partial z}(\frac{\partial \eta}{\partial t}) - \frac{\partial \phi}{\partial t}\right\}^2 + c_4\left(\frac{\partial^2 \phi}{\partial z \partial t}\right)^2\right] dz,$$

where c_1, c_2 are coefficients of external damping and c_3, c_4 are coefficients of internal (material) damping.

The modified Timoshenko equations are found after carrying out the ususal variational calculations [27] yielding

$$\rho A \frac{\partial^2 \eta}{\partial^2 t} = \frac{\partial}{\partial z}\left[\kappa G A\left(\frac{\partial \eta}{\partial z} - \phi\right)\right] - c_1\frac{\partial \eta}{\partial t} + c_3\left(\frac{\partial^3 \eta}{\partial z^2 \partial t} - \frac{\partial^2 \phi}{\partial z \partial t}\right) \tag{73}$$

$$\begin{aligned} \rho I \frac{\partial^2 \phi}{\partial t^2} &= \frac{\partial}{\partial z}\left[EI\frac{\partial \phi}{\partial z}\right] + \kappa G A\left(\frac{\partial \eta}{\partial z} - \phi\right) \\ &\quad -c_2\frac{\partial \phi}{\partial t} + c_3\left(\frac{\partial^2 \eta}{\partial z \partial t} - \frac{\partial \phi}{\partial t}\right) + c_4\frac{\partial^3 \phi}{\partial z^2 \partial t} \end{aligned} \tag{74}$$

To obtain the Bernoulli-Euler model we invoke the approximations $\rho I \to 0$ and $\partial\eta/\partial z - \phi \to 0$ with the result

$$\rho A\ddot{\eta} + c_4\frac{\partial^4 \dot{\eta}}{\partial z^4} - c_2\frac{\partial^2 \dot{\eta}}{\partial z^2} + c_1\dot{\eta} + EI\frac{\partial^4 \eta}{\partial z^4} = 0. \tag{75}$$

Table 1 summarizes the dissipation terms appearing in (75) in terms of standard damping mechanisms [51,54]. Individually, these dissipation types produce well known eigenvalue patterns for the spectrum of the abstract evolution operator A in (7). In particular, the viscous term translates the spectrum parallel to the real axis, 'square-root' damping gives the wedge pattern characteristic of material dissipation and Kelvin-Voight damping provides a pattern where the sequence of eigenvalues have an accumulation point located on the finite real axis [10]. See for example Pichè [53], Gibson [10], Chen and Russell [54], Wie and Bryson [30], and Balas [9].

Equations (73)–(74) are easily put in the first-order form appropriate for our standard control modeling problem. For the Timeshenko model, introduce the variables $\nu(t,z)$, $\gamma(t,z)$ and write the equations as

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= \frac{\partial \nu}{\partial z} + \frac{c_3}{\rho A}\frac{\partial^2 \eta}{\partial z^2} - \frac{c_3}{\rho A}\frac{\partial \phi}{\partial z} - c_1\eta, \\ \frac{\partial \nu}{\partial t} &= \frac{\kappa G}{\rho}\left(\frac{\partial \eta}{\partial z} - \phi\right), \end{aligned} \tag{76}$$

$$\begin{aligned}\frac{\partial \phi}{\partial t} &= \frac{\partial \gamma}{\partial z} + \frac{A}{I}\nu + \frac{c_4}{\rho I}\frac{\partial^2 \phi}{\partial z^2} + \frac{c_3}{\rho I}\frac{\partial \eta}{\partial z} - \frac{(c_2+c_3)}{\rho I}\phi, \\ \frac{\partial \gamma}{\partial t} &= \frac{E}{\rho}\frac{\partial \phi}{\partial z}.\end{aligned}$$

For the Bernoulli-Euler model (75), introduce the variable $\gamma(z,t)$ and write as

$$\begin{aligned}\frac{\partial \eta}{\partial t} &= -\frac{I}{A}\frac{\partial^2 \gamma}{\partial z^2} - \frac{c_4}{\rho A}\frac{\partial^4 \eta}{\partial z^4} + \frac{c_2}{\rho A}\frac{\partial^2 \eta}{\partial z^2} - c_1 \eta, \\ \frac{\partial \gamma}{\partial t} &= \frac{E}{\rho}\frac{\partial^2 \eta}{\partial z^2}.\end{aligned} \tag{77}$$

3.3 Frequency Response Calculations for DP Systems

In this section we will be concerned with the computation of certain irrational transfer functions and a resolvent operator. This provides a complete model including transient response for the DP system. For our purposes, the resolvent can be considered as an integral operator with kernel a *Green's function.* Using transform methods we compute explicit formulae for the abstract objects discussed previously. We focus on hyperbolic and parabolic linear (one dimensional) structural models for distributed elements. Such models can be used for elastic dynamics of beams, cables, etc. Finally, the computations are extended to hybrid system models consisting of interconnections of elastic components with rigid bodies and other lumped parameter models.

3.3.1 Hyperbolic Models

Consider a class of elastic structures represented by hyperbolic partial differential equations in one space dimension $0 \leq z \leq L$, (e.g. arising from models such as discussed in the previous section):

$$\frac{\partial x(t,z)}{\partial t} = F\frac{\partial x(t,z)}{\partial z} + Hx(t,z) + Ev(t,z) \tag{78}$$

subject to boundary conditions

$$\Sigma_1 x(t,0) + \Gamma_1 x(t,L) = Df(t), \tag{79}$$

and initial conditions

$$x(0,z) = x^0(z) \in \mathcal{H}^n(0,L). \tag{80}$$

Here, x is an n-vector valued state $x \in \mathcal{H}^n(0,L)$, $v \in \mathcal{H}^\ell(0,L)$ is an ℓ-vector valued distributed disturbance, f is m-vector valued boundary interactions, F, H are real $n \times n$ matrices with F nonsingular and diagonalizable [43], and Σ_1, Γ_1 are $n \times n$ matrices. Controllability questions for systems of this type are considered in Russell [15]. We remark that (81) is a concrete example of the transform of the abstract formula (9). After taking Laplace transforms in the temporal variable t, we obtain

$$\hat{X}(s,z) = \int_0^L G_r(s,z,w)\hat{M}(s,w)dw + H_{BC}(s,z)\hat{F}(s) \tag{81}$$

where

$$\hat{M}(s,z) = x^0(w) - C\,\hat{V}(s,w),$$

and $\hat{X}$, $\hat{V}$, $\hat{F}$ are the Laplace transforms of x, v, f respectively. The function $G_r(s,z,w)$ is the *Green's function* [50,52] for (78), (79) and $H_{BC}(s,z)$ is a transfer function from boundary interactions to state. Since in most cases of practical interest the control of flexible structures will be effected by actuators whose influence functions are highly localized, we have formulated our model with boundary control only.

Comparison with (1) clearly shows that the resolvent for the operator $A : \mathcal{H}^n(0,L) \rightarrow \mathcal{H}^n(0,L)$ defined by (78) and (79) is the integral operator $\int_0^L G_r(s,z,w) \cdot dw$.

A straightforward calculation leads to the following form for H_{BC}

$$H_{BC}(s,z) = N(s,z)D \tag{82}$$

where

$$\begin{aligned} \Phi(s,z) &= e^{[F^{-1}(sI-H)z]}, \\ N(s,z) &= \Phi(s,z)\left[\Sigma_1 + \Gamma_1\Phi(s,L)\right]^{-1} \end{aligned} \tag{83}$$

The Green's function for (78), (79) is the solution to

$$\frac{\partial G_r(s,z,w)}{\partial z} = F^{-1}\left[sI - H\right]G_r(s,z,w) + I_n\delta(z-w) \tag{84}$$

subject to the boundary conditions

$$\Sigma_1 G_r(s,0,w) + \Gamma_1 G_r(s,L,w) = 0 \tag{85}$$

where $\delta(\cdot)$ is the Dirac delta function [50], [52]. From (84) we see that the solution is discontinuous at the point $z = w$. After some computation, we can write

$$G_r(s,z,w) = \begin{cases} G_{r\,LEFT}(s,z,w), & \text{for } 0 \le z \le w \\ G_{r\,RIGHT}(s,z,w), & \text{for } w \le z \le L \end{cases} \tag{86}$$

with

$$G_{r\,LEFT}(s,z,w) = -N(s,z)\Gamma_1\Phi(s,L-w), \tag{87}$$

$$G_{r\,RIGHT}(s,z,w) = N(s,z)\Sigma_1\Phi(s,-w). \tag{88}$$

The required calculations are now summarized. To obtain the forced response to boundary control $f(t)$ can be found in the Laplace transform domain as the solution to

$$\frac{d}{dz}\hat{x}(s,z) = F^{-1}(sI - H)\hat{x}(s,z)$$

subject to boundary conditions

$$\Sigma_1\hat{x}(s,0) + \Gamma_1\hat{x}(s,L) = D\hat{f}(s).$$

The general solution is

$$\hat{x}(s,z) = e^{F^{-1}(sI-H)z}k$$

where k is an n-vector to be determined. Substituting the general solution into the boundary conditions yields

$$\Sigma_1 k + \Gamma_1 e^{F^{-1}(sI-H)L}k = D\hat{f}(s)$$

or

$$k = [\Sigma_1 + \Gamma_1 e^{F^{-1}(sI-H)L}]^{-1} D\hat{f}(s).$$

By substituting k into the general solution the terms in (82)–(83) can be recognized.

Derivation of the Green's function follows from (84)–(85) as follows. Note that integrating (84) over a small interval $w - \epsilon \leq z \leq w + \epsilon$ gives

$$\int_{w-\epsilon}^{w+\epsilon} \frac{d}{dz} G_r(s;z,w)dz = \int_{w-\epsilon}^{w+\epsilon} F^{-1}(sI-H)G_r(s;z,w)dz + I_n$$

by the sifting property of the delta function. Then by letting $\epsilon \to 0$ we see that the second term on the right hand side vanishs and we get

$$G_r(s;w_+,w) - G_r(s;w_-,w) = I_n, \tag{89}$$

so the Green's function is discontinuous at $z = w$. A general solution to (84) has the form (86) with

$$\begin{aligned} G_{r\,LEFT}(s,z,w) &= e^{F^{-1}(sI-H)z}K_\ell \\ G_{r\,RIGHT}(s,z,w) &= e^{F^{-1}(sI-H)z}K_r, \end{aligned}$$

where K_ℓ, K_r are $n \times n$ matrices to be determined in the sequel. Substituting into (89) gives

$$K_r - K_\ell = e^{-F^{-1}(sI-H)w} \tag{90}$$

and into (85) gives

$$\Sigma_1 K_\ell + \Gamma_1 e^{F^{-1}(sI-H)L}K_r = 0. \tag{91}$$

Substitute K_r from (90) into (91) and solve for K_ℓ

$$K_\ell = -[\Sigma_1 + \Gamma_1 e^{F^{-1}(sI-H)L}]^{-1}\Gamma_1 e^{F^{-1}(sI-H)(L-w)}.$$

Now substituting K_ℓ into (90) and solving for K_r gives

$$\begin{aligned} K_r &= e^{-F^{-1}(sI-H)w} - [\Sigma_1 + \Gamma_1 e^{F^{-1}(sI-H)L}]^{-1}\Gamma_1 e^{F^{-1}(sI-H)(L-w)} \\ &= \left[I - [\Sigma_1 + \Gamma_1 e^{F^{-1}(sI-H)L}]^{-1}\Gamma_1 e^{F^{-1}(sI-H)L}\right] e^{-F^{-1}(sI-H)w} \\ &= [\Sigma_1 + \Gamma_1 e^{F^{-1}(sI-H)L}]^{-1}\left[\Sigma_1 + \Gamma_1 e^{F^{-1}(sI-H)L} - \Gamma_1 e^{F^{-1}(sI-H)L}\right] e^{-F^{-1}(sI-H)w} \\ &= [\Sigma_1 + \Gamma_1 e^{F^{-1}(sI-H)L}]^{-1}\Sigma_1 e^{-F^{-1}(sI-H)w}. \end{aligned}$$

Substitution then into the general solution gives the forms in (87)–(88).

3.3.2 Parabolic Models

We begin with the canonical, first-order model consisting of n equations as

$$\frac{\partial x(t,z)}{\partial t} = G\frac{\partial^2 x(t,z)}{\partial z^2} + F\frac{\partial x(t,z)}{\partial z} + Hx(t,z) + Ev(t,z) \tag{92}$$

subject to $2n$ boundary conditions

$$\Sigma_1 x(t,0) + \Sigma_2\frac{\partial x(t,0)}{\partial z} + \Gamma_1 x(t,L) + \Gamma_2\frac{\partial x(t,L)}{\partial z} = Df(t), \tag{93}$$

and n initial conditions

$$x(0,z) = x^0(z) \in \mathcal{H}^n(0,L). \tag{94}$$

Let $\Sigma = [\Sigma_1, \Sigma_2]$ and $\Gamma = [\Gamma_1, \Gamma_2]$—each $2n \times 2n$ real matrics.

Following a procedure as above the Green's function and boundary transfer function can be computed as follows. Let

$$\begin{aligned} \Lambda(s) &\equiv \begin{bmatrix} 0_n & I_n \\ -G^{-1}(H - sI_n) & -G^{-1}F \end{bmatrix}, \\ \Phi(s,z) &\equiv e^{\Lambda(s)z}, \qquad (95) \\ M(s,z) &= [I_n, 0]\Phi(s,z)[\Sigma + \Gamma\Phi(s,L)]^{-1}. \qquad (96) \end{aligned}$$

Then the boundary transfer function is

$$H_{BC}(s,z) = M(s,z)D, \tag{97}$$

and the Green's function is given by

$$G_r(s,z,w) = \begin{cases} G_{r\,LEFT}(s,z,w), & \text{for } 0 \le z \le w \\ G_{r\,RIGHT}(s,z,w), & \text{for } w \le z \le L \end{cases} \tag{98}$$

with

$$\begin{aligned} G_{r\,LEFT} &= -M(s,z)\Gamma\Phi(s,L-w)\begin{bmatrix} 0 \\ I_n \end{bmatrix}, \qquad (99) \\ G_{r\,RIGHT} &= M(s,z)\Sigma\Phi(s,-w)\begin{bmatrix} 0 \\ I_n \end{bmatrix}. \end{aligned}$$

3.4 Modeling of Hybrid Systems

In most applications, models for the dynamics of flexible structures will involve interaction between various elastic and rigid elements. In the particular case of flexible structures associated with large space structures, the potential topological configurations can be quite complex. Various elements such as beams, truss structures, cables, membranes, etc., may have dominant distributed parameter effects. Typically a central body or bodies represent large concentrations of mass with respect to the overall low mass density of the

flexible structure. These are most effectively represented by lumped parameter models of their rigid body dynamics. Additionally, various attitude control actuators can add concentrated inertia elements which can be effectively modeled as lumped systems. Thus, carefully chosen linear, hybrid models can provide an effective tool for analysis of dynamics of vibrations and their effect on small angle motions for complex space platforms. In this section, we consider the structures and computations of certain resulting transfer functions and the resolvent operator for the composite system along the lines of Section 1.

The concept of a mechanical impedance (terminology borrowed from electrical network theory) has been used in structural dynamic modeling for many years [51]. The dynamic stiffness method (application to space structure modeling is reviewed in Pichè [53]) uses this notion to compute effective transfer function models for interconnected structures [55]. Our approach here will follow along similar lines except that we will focus on computing the resolvent operator for a hybrid structure by direct manipulation of its kernel; viz, a Green's function [52]. A hybrid, state space model is constructed in Burns and Cliff [12] (where considerations are given for approximation and computation in the hybrid state space). We will consider a hybrid state space as consisting of a direct sum of spaces $\mathcal{X} = \mathcal{X}_\ell \oplus \mathcal{X}_d$ where $\mathcal{X}_d = \mathcal{H}^{N_d}$ is the distributed part constructed on an appropriate Hilbert space of N_d-vector valued functions with the "energy" inner product of (6) for a distributed parameter system (DPS) written in abstract form (7) and $\mathcal{X}_\ell \equiv \Re^{N_\ell}$, a finite dimensional state space of the lumped parameter system (LPS).

For control of hybrid structures, we restrict attention to DPS modeled as either the hyperbolic or parabolic (or mixed) cases which, as we have seen, can be expressed in the frequency domain in the form

$$\hat{X}_d(s,z) = \int_0^L G_r(s,z,w)\hat{M}(s,w)dw + H_{BC}(s,q)\hat{F}_d(s) \tag{100}$$

where

$$\hat{M}(s,w) = x_d^0(w) - E\hat{V}(s,w). \tag{101}$$

Clearly, (100)-(101) can represent a disjoint collection of distributed elements such as beams, cables, etc. (Conceptually, a version of (100) can also be written for higher dimensional spatial domains, but we feel for the current presentation that the required complexity of notation can mask the simplicity of the underlying concepts—see Butkovskiy for details [52].)

All LPS component models are combined into a LPS state space model as

$$\dot{x}_\ell(t) = A_\ell x_\ell(t) + B_\ell f_\ell(t),\ x_\ell^o = x_\ell(0) \tag{102}$$

with $x_\ell \in R^{N_\ell} \equiv \mathcal{X}_\ell$ a finite dimensional real space. By taking Laplace transforms in (102), we write (analogous to (100))

$$\hat{X}_\ell(s) = R_\ell(s)x_\ell^0 + H_\ell(s)\hat{F}_\ell(s), \tag{103}$$

where $R_\ell(s) = [sI_{N_\ell} - A_\ell]^{-1}$ is the resolvent for the (matrix) operator A_ℓ and $H_\ell(s) = R_\ell(s)B$.

The hybrid state space $\mathcal{X} = \mathcal{X}_\ell \oplus \mathcal{X}_d$ consists of elements

$$x(t,z) = \begin{pmatrix} x_\ell(t) \\ x_d(t,z) \end{pmatrix} \tag{104}$$

which are $N = N_d + N_\ell$-valued functions of $z \in [0,L]$, $t > 0$. Finally, the interconnection of component systems is resolved through a topological constraint relation consisting of $m = m_d + m_\ell$ linear equations;

$$f(t) + T_1 x_d(t,0) + T_2 x_d(t,L) + T_3 x_\ell(t) = Ku(t) \tag{105}$$

where $u(t)$ is a k-vector of control inputs to the hybrid system, T_1, T_2 are $m \times N_d$, T_3 is $m \times N_\ell$, and K is $m \times k$ real matrices. The hybrid modeling problem is to find an equation of the form (100) by solving (100), (103)–(105) simultaneously for the hybrid state $x(t,z)$. We provide the resulting model in the following form:

$$\hat{X}(s,z) = \int_0^L \tilde{G}_r(s,z,w)\hat{M}(s,w)dw + \tilde{R}(s,z)x_\ell^0 + \tilde{H}_{BC}(s,z)\hat{U}(s), \tag{106}$$

where $\hat{M}(s,w)$ is given in (101). The resolvent operator for the hybrid system is

$$R(s;\mathbf{A}) = \left[\tilde{R}(s,z), \int_0^L \tilde{G}_r(s,z,w) \cdot dw \right] \tag{107}$$

where $R(s;\mathbf{A}) : \mathcal{X} \to D(\mathbf{A}) \subseteq \mathcal{X}$, $\tilde{G}_r$ is $N \times N_d$ and $\tilde{R}$ is $N \times N_\ell$ are matrix valued functions which can be computed explicitly as follows:

$$\tilde{R}(s,z) = \begin{bmatrix} I_{N_\ell} - H_\ell(s)\tilde{Q}_1(s) \\ -H_{BC}(s,z)\tilde{Q}_2(s) \end{bmatrix} T_3 R_\ell(s), \tag{108}$$

$$\tilde{G}_r(s,z,w) = \begin{bmatrix} -H_\ell(s)\tilde{Q}_1(s) \\ G_r(s,z,w) - H_{BC}(s,z)\tilde{Q}_2(s) \end{bmatrix} P(s,w) \tag{109}$$

where

$$\tilde{Q}(s) = [I_M + Q(s)]^{-1} = \begin{bmatrix} \tilde{Q}_1(s) \\ \tilde{Q}_2(s) \end{bmatrix}, \tag{110}$$

$$Q(s) = [T_3 H_\ell(s),\ T_1 H_{BC}(s,0) + T_2 H_{BC}(s,L)], \tag{111}$$

$$P(s,w) = T_1 G_r(s,0,w) + T_2 G_r(s,L,w). \tag{112}$$

Finally, the $N \times k$ transfer function matrix from boundary control to hybrid state is

$$\tilde{H}(s,z) = \begin{bmatrix} H_\ell(s) & 0 \\ 0 & H_{BC}(s,z) \end{bmatrix} \tilde{Q}(s)K. \tag{113}$$

The derivation of (106)–(112) is straightforward and proceeds as follows. Substitute (100), (102) into (105) and solve for the interconnecting force $\hat{F}(s)$. This identifies the terms $Q(s), P(s,w)$ above. Now substitute the appropriate components of $\hat{F}(s)$ into (100), (102) and use the hybrid state model (104).

In Section 5 of this report we consider several examples illustrating these calculations.

4 Homogenization of regular structures

In this section we consider the problem of modeling and control of a class of lattice structures, e.g., trusses. Their large size and repetitive infrastructure require special techniques for structural analysis to cope with the large number of degrees of freedom. Approximations of such systems by continua provide a simple means for comparing structural characteristics of lattices with different configurations, and they are effective in representing macroscopic vibrational modes and structural response due to temperature and load inputs. Our approach to the construction of such models is based on a technique for asymptotic analysis called *homogenization*. It has been widely used in mathematical physics for the treatment of composite systems like porous media for which one wishes to have an effective approximating system with parameters which are constant across the structure.[1]

Before developing the general features of the method and applying it to the treatment of lattice structures, we shall make a few remarks on other work on continuum models which has appeared in the recent structural mechanics literature.

Noor, et. al. [1] use an energy method to derive a continuum approximation for trusses with triangular cross sections in which the modal displacements of the truss are related to a linearly varying displacement field for an equivalent bar. Plates with a lattice infrastructure are also treated. In Dean and Tauber [64] and Renton [74] exact analytical expressions for the solutions of trusses under load were derived using finite difference calculus. By expressing the difference operators in terms of Taylor's series Renton [77] was able to derive continuum approximations to the finite difference equations resulting in expressions for equivalent plate stiffnesses, for example. In a recent paper Renton [77] used this approach to give equivalent beam properties for trusses, which complements the earlier work of Noor, Anderson and Greene [3], and Nayfeh and Hefzy [2]. (See also (Anderson [3]).)

In most cases a continuum model is associated with the original (lattice) structure by averaging the parameters of the lattice over some natural volume (e.g., of a "cell" of the structure) and identifying the averaged parameter value (mass density, stress tensor, etc.) with the corresponding distributed parameter in the continuum model. A specific form for the continuum model is postulated at the outset of the analysis; e.g., a truss with lattice structure will be approximated by a beam, with the beam dynamical representation assumed in advance. While this approach has an appealing directness and simplicity, it has some problems.

First, it is very easy to construct an example in which the "approximate model" obtained by averaging the parameters over a cell is not a correct approximation to the system behavior. This is done in subsection 4.1.[2] Second, one cannot use this procedure to obtain "corrections" to the approximation based on higher order terms in an expansion, which may sometimes be done in an asymptotic analysis. These terms can be used to describe the microscopic behavior (e.g., local stresses) in the structure. Third, the averaging method

[1] See, for example, the papers of Larsen [69], Keller [68], and the reports of Babuska [57] for applications and discussions of design techniques.

[2] See the numerical experiments in (Bourgat [62]).

(averaging the parameters over space) does not apply in a straightforward way to systems with a random structure, since the appropriate averaging procedure may not be obvious.[3] Fourth, the method cannot be naturally imbedded in an optimization procedure; and controls and state estimates based on the averaged model may not be accurate reflections of controls and state estimates derived in the course of a unified optimization - averaging procedure. In particular, the method does not provide a systematic way of estimating the degree of suboptimality of controls and state estimates computed from the idealized model.

In this work we use a totally different technique called *homogenization* from the mathematical theory of asymptotic analysis to approximate the dynamics of structures with a repeating cellular structure. Homogenization produces the distributed model as a consequence of an asymptotic analysis carried out on a rescaled version of the physical system model.

Unlike the averaging method, homogenization can be used in combination with optimization procedures; and it can yield systematic estimates for the degree of suboptimality of controls and estimators derived from idealized models. While our results are stated in terms of simple structures, they demonstrate the feasibility of the method; and they suggest its potential in the analysis of structures of realistic complexity.

In subsection 4.1 we give an example derived from (Bensoussan, Lions, and Papanicolaou [60]) illustrating some of the subtleties of homogenization, particularly in the context of control problems. In subsection 4.2 we derive a homogenized representation for the dynamics of a lattice structure undergoing transverse deflections. We show that the behavior of the lattice is well approximated by the Timenshenko beam equation; and we show that this equation arises naturally as the limit of the lattice dynamics when the density of the lattice structure goes to infinity in a well defined way. The problem of vibration control of a lattice is posed and discussed in subsection 4.3. In subsection 4.4 we derive a diffusion approximation for the thermal conductivity of a one-dimensional lattice structure. This property is useful in analyzing new materials for large space structures.

Acknowledgements: We are grateful to Professor George Papanicolaou for bringing Kunnemann's paper to our attention and to Drs. A. Amos and R. Lindberg for their comments on an earlier version of this work.

4.1 A one-dimensional example

From (Bensoussan, Lions, and Papanicolaou [60]) we have the following example:

$$-\frac{d}{dx}[a^\epsilon(x)\frac{du^\epsilon(x)}{dx}] = f(x), x \in (x_0, x_1) \tag{114}$$

$$u^\epsilon(x_0) = 0 = u^\epsilon(x_1)$$

[3] Homogenization methods do apply to systems with a randomly heterogeneous structure, see (Papanicolaou and Varadhan [71]) and (Kunnemann [78]). We shall treat such systems in a subsequent report.

where $a^\epsilon(x) \stackrel{\text{def}}{=} a(x/\epsilon)$, and $a(y)$ is periodic in y with period Y_0, $a(y) \geq \alpha > 0$. It is simple to show that

$$||u^\epsilon||^2_{H^1} \stackrel{\text{def}}{=} \int_{x_0}^{x_1} |u^\epsilon(x)|^2 + |\frac{du^\epsilon(x)}{dx}|^2 dx \leq c \tag{115}$$

and so, $u^\epsilon \rightarrow u$ weakly in the Hilbert space H^1.[4] Moreover,

$$a^\epsilon \rightarrow M(a) \stackrel{\text{def}}{=} \frac{1}{Y_0} \int_0^{Y_0} a(y) dy \tag{116}$$

and it is natural to suppose that $u^\epsilon \rightarrow u$ with the limit defined by

$$-\frac{d}{dx}[M(a)\frac{d}{dx}u(x)] = f(x), x \in (x_0, x_1) \tag{117}$$

$$u(x_0) = u(x_1)$$

This is untrue in general (Bensoussan, Lions, and Papanicolaou [60], pp. 8-10). The correct limit is given by

$$-\frac{d}{dx}[\bar{a}\frac{d}{dx}u(x)] = f(x), x \in (x_0, x_1) \tag{118}$$

$$u(x_0) = u(x_1)$$

with

$$\bar{a} \stackrel{\text{def}}{=} [M(\frac{1}{a})]^{-1} \tag{119}$$

In general, $M(a) > \bar{a}$; and so, the error in identifying the limit, (117) versus (118), is fundamental.

The system (117) corresponds to averaging the parameter $a^\epsilon(x)$ over a natural *cell;* a procedure similar to that used in the past to define continuum models for lattice structures. As (118) shows, the actual averaging process can be more subtle than one might expect, even for simple problems.

4.1.1 Homogenization of the example

To see how (118) arises, we can use the method of *multiple scales* which applies to a variety of perturbation problems. Suppose

$$u^\epsilon(x) = u^\epsilon(x, \frac{x}{\epsilon}) = u_0(x, \frac{x}{\epsilon}) + \epsilon u_1(x, \frac{x}{\epsilon}) + \ldots \tag{120}$$

that is, we suppose that u^ϵ depends on the "slow" scale x and the "fast" scale $y \stackrel{\text{def}}{=} x/\epsilon$; and we adopt an *ansatz* which reflects this dependence. Using the identity

$$\frac{d}{dx}[u(x, \frac{x}{\epsilon})] = \frac{\partial u}{\partial x} + \frac{1}{\epsilon}\frac{\partial u}{\partial y}, y = \frac{x}{\epsilon} \tag{121}$$

[4] Here $H^1 \stackrel{\text{def}}{=} \{u \in L^2(x_0, x_1) : ||u||_{H^1} < \infty\}$

then (114) may be rewritten as

$$-(\frac{\partial}{\partial x}+\frac{1}{\epsilon}\frac{\partial}{\partial y})\{a(y)(\frac{\partial}{\partial x}+\frac{1}{\epsilon}\frac{\partial}{\partial y})[u_0+\epsilon u_1+...]\}=f \tag{122}$$

Simplifying and equating coefficients of like powers of ϵ, we find first that

$$-\frac{1}{\epsilon^2}\frac{\partial}{\partial y}[a(y)\frac{\partial}{\partial y}u_0]=0. \tag{123}$$

The assumptions on *a(y)* imply

$$u_0(x,y)=u_0(x) \tag{124}$$

i.e., no y-dependence. The coefficients of ϵ^{-1} satisfy

$$\{\frac{\partial}{\partial y}[a(y)\frac{\partial}{\partial x}u_0]+\frac{\partial}{\partial x}[a(y)\frac{\partial}{\partial y}u_0]-\frac{\partial}{\partial y}[a(y)\frac{\partial}{\partial y}u_1]\}=0 \tag{125}$$

or

$$\frac{\partial}{\partial y}[a(y)\frac{\partial}{\partial y}u_1]=-\frac{\partial a}{\partial y}\frac{\partial u_0}{\partial x} \tag{126}$$

If we look for u_1 in the form

$$u_1(x,y)=-\chi(y)\frac{\partial u_0}{\partial x}+\hat{u}_1(x), \tag{127}$$

then the *corrector* $\chi(y)$ must satisfy

$$-\frac{d}{dy}[a(y)\frac{d}{dy}\chi(y)]=-\frac{da}{dy} \tag{128}$$

and be periodic. That is,

$$a(y)\frac{d\chi}{dy}=a(y)+c \tag{129}$$

which has a periodic solution (unique up to an additive constant in y) if and only if

$$\frac{1}{Y_0}\int_0^{Y_0}[1+\frac{c}{a(y)}]dy=0 \tag{130}$$

which implies

$$c=-[M(\frac{1}{a})]^{-1}\stackrel{\text{def}}{=}\bar{a} \tag{131}$$

We obtain an equation for $u_0(x)$ from the solvability condition for $u_2(x,y)$. Equating the coefficients of ϵ^0 in the expansion, we have

$$-\frac{\partial}{\partial y}[a(y)\frac{\partial}{\partial y}u_2]-\frac{\partial}{\partial y}[a(y)\frac{\partial}{\partial x}u_1] \tag{132}$$

$$-\frac{\partial}{\partial x}[a(y)\frac{\partial}{\partial y}u_1]-\frac{\partial}{\partial x}[a(y)\frac{\partial}{\partial x}u_0]=f(x)$$

This has a solution $u_2(x,y)$, periodic in y if and only if

$$\{\frac{1}{Y_0}\int_0^{Y_0}[a(y)+\frac{\partial}{\partial y}[a(y)\chi(y)]-a(y)\frac{\partial}{\partial y}\chi(y)]dy\}\cdot\frac{d^2u_0(x)}{dx^2} \tag{133}$$

$$+f(x)=0$$

where we have used (127). The integral of the second term is zero, since it is the integral of the derivative of a periodic function over one period. Using (129) and (131), (133) reduces to

$$-\bar{a}\frac{d^2u_0}{dx^2}+f(x)=0 \tag{134}$$

(plus the boundary conditions) which is (118) (119).

4.1.2 Control and homogenization of the one dimensional system

One of the simplest stochastic control problems associated with the preceding system is defined by the Hamilton - Jacobi - Bellman equation

$$-a(\frac{x}{\epsilon})\frac{d^2u^\epsilon}{dx^2}-\frac{1}{\epsilon}b(\frac{x}{\epsilon})\frac{du^\epsilon}{dx}$$

$$=\inf_{v\in\mathcal{R}}[\frac{1}{2}v^2+g(x,y)v\frac{du^\epsilon}{dx}-cu^\epsilon] \tag{135}$$

$$x\in\mathbf{O},u^\epsilon(x)=0 on\Gamma\stackrel{\text{def}}{=}\partial\mathbf{O}$$

where $\mathbf{O}$ is an open interval in $\mathcal{R}$, and each function $a(y),b(y)$, and $g(x,y)$ is periodic in y with period Y_0. We assume that $a(y)\geq\alpha>0$ and that $c>0$, and that the controls v take values in $\mathcal{R}$.

This Bellman equation corresponds to the stochastic control problem

$$u^\epsilon(x)=\inf_{v(\cdot)}J^\epsilon[v(\cdot)]$$

$$J^\epsilon[v(\cdot)]=E_x\{\int_0^{\tau_\alpha}l(x^\epsilon,\frac{x^\epsilon}{\epsilon},v)[e^{-\int_0^t c(x^\epsilon,\frac{x^\epsilon}{\epsilon},v)ds}]dt\} \tag{136}$$

$$dx^\epsilon(t)=\sigma(x^\epsilon,\frac{x^\epsilon}{\epsilon})dw(t)+\frac{1}{\epsilon}b(x^\epsilon,\frac{x^\epsilon}{\epsilon})dt+G(x^\epsilon,\frac{x^\epsilon}{\epsilon},v)dt$$

$$x^\epsilon(0)=x\in\mathbf{O},t\geq 0.$$

with $\sigma^2(x,y)\stackrel{\text{def}}{=}a(y)$, $b(x,y)=b(y)$, $G(x,y,v)=g(x,y)v$, $l(x,y,v)=\frac{1}{2}v^2$, and $c(x,y,v)=c$, a constant in (135). Each function in (136) is assumed to be periodic in y with period one. We are interested in the behavior of the optimal cost and control law for (135) in the limit as $\epsilon\to 0$. The stochastic control problem (136) was treated in (Bensoussan, Boccardo, and Murat [59]); the analysis here uses different arguments which emphasize the computational aspects of the system.

Evaluating the infimum in (135), we have the nonlinear system

$$a(\frac{x}{\epsilon})u^\epsilon_{xx} + \frac{1}{\epsilon}b(\frac{x}{\epsilon})u^\epsilon_x - cu^\epsilon_x - \frac{1}{2}g^2(x,\frac{x}{\epsilon})(u^\epsilon_x)^2 = 0 \tag{137}$$

$$x \in \mathbf{O}, u^\epsilon(x)|_\Gamma = 0.$$

The analysis of the control problem involves homogenization of this system.

Let

$$A_1 = a(y)\partial_{yy} + b(y)\partial_y \tag{138}$$

with its formal adjoint defined by

$$A_1^* = \partial_y[a(y)\partial_y \cdot] - \partial_y[(b(y) - a_y(y))\cdot]. \tag{139}$$

The problem

$$A_1^* m = 0, y \to m(y) \text{ periodic} \tag{140}$$

$$m > 0, \int_Y m(y)dy = 1$$

has a unique solution $m(\cdot)$ on $Y = S^0$, the unit circle, with

$$0 < \underline{m} \leq m(y) \leq \bar{m} < \infty. \tag{141}$$

So $m(\cdot)$ is a density on Y. We *assume* that $b(\cdot)$ is *centered*

$$\int_Y m(y)b(y)dy = 0. \tag{142}$$

As a consequence the system

$$A_1\chi(y) = b(y) \tag{143}$$

$$y \to \chi(y) \text{ periodic }, \int_Y \chi(y)dy = 0$$

has a well defined solution. $\chi(\cdot)$ is the *corrector* associated with the problem.

As before we set $y = x/\epsilon$ and look for u^ϵ in the form

$$u^\epsilon(x) = u^\epsilon(x,y) = u_0(x,y) + \epsilon u_1(x,y) + \ldots, \tag{144}$$

and we use

$$\partial_x\phi(x,y) = \phi x(x,y) + \frac{1}{\epsilon}\phi_y(x,y), y = x/\epsilon \tag{145}$$

$$\partial_{xx}\phi(x,y) = \phi_{xx}(x,y) + 2\epsilon\phi_{xy}(x,y) + \phi_{yy}(x,y).$$

Substituting in (137), we have

$$a(y)[u_{0xx} + 2\epsilon u_{0xy} + \frac{1}{\epsilon^2}u_{0yy}] + a(y)[\epsilon u_{1xx} + 2u_{1xy} + \frac{1}{\epsilon}u_{1yy}]$$

$$+a(y)[\epsilon^2 u_{2xx} + 2\epsilon u_{2xy} + u_{2yy}]$$

$$+ \frac{1}{\epsilon}b(y)[u_{0x} + \frac{1}{\epsilon}u_{0y}] + \frac{1}{\epsilon}b(y)[\epsilon u_{1x} + u_{1y}] \tag{146}$$

$$+\frac{1}{\epsilon}b(y)[\epsilon^2 u_{2x} + \epsilon u_{2y}] - c[u_0 + \epsilon u_1 + \epsilon^2 u_2]$$

$$-\frac{1}{2}g^2(x,y)[(u_{0x} + \epsilon u_{1x} + \epsilon^2 u_{2x}) + \frac{1}{\epsilon}(u_{0y} + \epsilon u_{1y} + \epsilon^2 u_{2y})]^2 = O(\epsilon^2)$$

The last term is

$$-\frac{1}{2}g^2(x,y)[(\frac{1}{\epsilon^2}u_{0y}^2 + 2\epsilon u_{0x}u_{0y} + 2u_{0x}u_{1y} + u_{0x}^2) \tag{147}$$

$$+\epsilon(2u_{0x}u_{1x} + 2u_{1x}u_{1y} + 2u_{1y}u_{2y} + 2u_{0x}u_{2y})] + O(\epsilon^2)$$

Equating coefficients of like powers of ϵ, we obtain

$$(\epsilon^{-2})a(y)u_{0yy} + b(y)u_{0y} - \frac{1}{2}g^2(x,y)u_{0y}^2 = 0 \tag{148}$$

$$(\epsilon^{-1})a(y)u_{1yy} + b(y)u_{1y} + 2a(y)u_{0xx} \tag{149}$$

$$+b(y)u_{0x} - g^2(x,y)u_{0x}u_{0y} = 0$$

$$(\epsilon^0)a(y)u_{2yy} + b(y)u_{2y} + 2a(y)u_{1xy} + b(y)u_{1x} \tag{150}$$

$$+a(y)u_{0xx} - cu_0 - \frac{1}{2}g^2(x,y)u_{0x}^2 - g^2(x,y)u_{0x}u_{1x} = 0.$$

Choosing $u_0(x,y) = u_0(x)$, which must be justified, satisfies (148). We can then solve (149) by choosing

$$u_1(x,y) = -\chi(y)u_{0x}(x) + \hat{u}_1(x). \tag{151}$$

Equation (150) has a solution for $u_2(x,y)$ if

$$\int_Y m(y)\{-2a(y)\chi_y u_{0xx} - b(y)\chi_y u_{0x} + a(y)u_{0xx} \tag{152}$$

$$-cu_0 - \frac{1}{2}g^2(x,y)(1-2\chi_y)u_{0x}^2\}dy = 0.$$

This gives an equation for $u_0(x)$

$$qu_{0xx} - cu_0 - \frac{1}{2}\Gamma u_{0x}^2 = 0$$

where

$$q \stackrel{\text{def}}{=} \int_Y m(y)\{a(y)[1-2\chi_y(y)] - \chi(y)b(y)\}dy$$

$$\Gamma \stackrel{\text{def}}{=} \int_Y m(y)g^2(x,y)[1-2\chi_y(y)]dy.$$

Remark. From the definition of A_1 and the corrector $\chi(y)$ we have

$$\int_Y m(y)b(y)\chi(y)dy = \int_Y [a(y)\chi_{yy} + b(y)\chi_y]\chi(y)m(y)dy \tag{155}$$

$$= \int_Y \chi(y)\partial_{yy}[a(y)\chi(y)m(y)]dy - \int_Y \chi(y)\partial_y[b(y)\chi(y)m(y)]dy$$

Also, using (143),

$$\int_Y m(y)b(y)\chi(y)dy = \int_Y \chi(y)a(y)m(y)\chi_{yy}(y)dy \tag{156}$$

$$-\int_Y \chi(y)b(y)m(y)\chi_y dy + 2\int_Y \chi(y)\chi_y \partial_y[a(y)m(y)]dy$$

Adding these two expressions, we have

$$2\int_Y m(y)b(y)\chi(y)dy$$

$$= 2\int_Y \chi(y)a(y)m(y)\chi_{yy}dy + 2\int_Y \chi(y)\chi_y[am]_y dy \tag{157}$$

$$= -2\int_Y \partial_y[\chi am]\chi_y dy + 2\int_Y \chi(y)\chi_y[am]_y dy - 2\int_Y \chi_y a(y)m(y)\chi_y dy$$

Thus, q may be rewritten as

$$q = \int_Y m(y)\{a(y)[1 - 2\chi_y + \chi_y^2]\}dy \tag{158}$$

$$= \int_Y m(y)a(y)[1 - \chi_y]^2 dy$$

and clearly $q \geq 0$.

The term q in (154) summarizes the effects of the averaging process on the uncontrolled system. The homogenization process interacts with the control system through the term Γ, whose form would be difficult to "guess" from simple averaging procedures.

4.2 Continuum Model for a Simple Structural Mechanical System

4.2.1 Problem definition

Consider the truss shown in Figure 1 (undergoing an exaggerated deformation). We shall assume that the truss has a regular (e.g., triangular) cross-section and no "interlacing" supports. We assume that the displacements of the system are "small" in the sense that no components in the system buckle. We are interested in describing the dynamical behavior of the system when the number of cells (a unit between two (triangular) cross sections) is large; that is, in the limit as

$$\epsilon \stackrel{\text{def}}{=} \ell/L \to 0. \tag{159}$$

We shall make several assumptions to simplify the analysis. First, we shall assume that the triangular sections are essentially rigid, and that all mobility of the system derives from the flexibility of the members connecting the triangular components. Second, we shall ignore damping and frictional effects in the system. Third, we shall confine attention to small transverse displacements $\eta(t,x)$ and small in plane rotations $\phi(t,x)$ as indicated in Figure 1, ignoring longitudinal and out of plane motions and torsional twisting. Fourth,

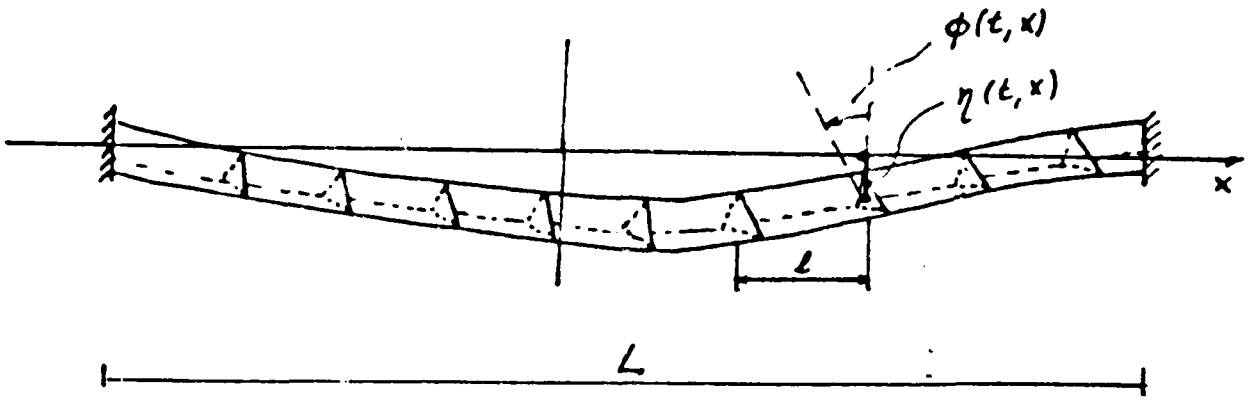


Figure 1: Deformed truss with regular cross-section.

we shall assume that the mass of the triangular cross members dominates the mass of the interconnecting links.

Systems of this type have been considered in several papers including [1], [2], [3], and [77]. In those papers a continuum beam model was hypothesized and effective values for the continuum system parameters were computed by averaging the associated parameters of the discrete system. Our approach to the problem is based on homogenization – asymptotic analysis and is quite different.

The assumptions simplify the problem substantially, by suppressing the geometric structure of the truss. We can retain this structure by writing dynamical equations for the nodal displacements of the truss members. For triangular cross sections nine parameters describe the displacements of each sectional element. The analysis which follows may be carried over to this case, but the algebraic complexity prevents a clear presentation of the main ideas. As suggested in (Noor et al. [1]) one should use a symbolic manipulation program like MACSYMA to carry out the complete details of the calculations. We shall take up this problem on another occasion; for now we shall treat the highly simplified problem which, as we shall see, leads to the Timoshenko beam.

We shall begin by reformulating the system in terms of a discrete element model as suggested in (Crandal et al. [27]); see Figure 2. In this model we follow the displacement $\eta_i(t)$ and rotation $\phi_i(t)$ of the i^{th} mass M. The bending springs (k_b^i) tend to keep the system straight by keeping the masses parallel and the shearing springs (k_s^i) tend to keep the masses perpendicular to the connecting links. We assume small displacements and rotations so the approximations

$$\sin\phi_i(t) \cong \phi_i(t) \qquad (160)$$
$$\tan^{-1}[\eta_i(t)/\ell] \cong \eta_i(t)/\ell$$

are valid.

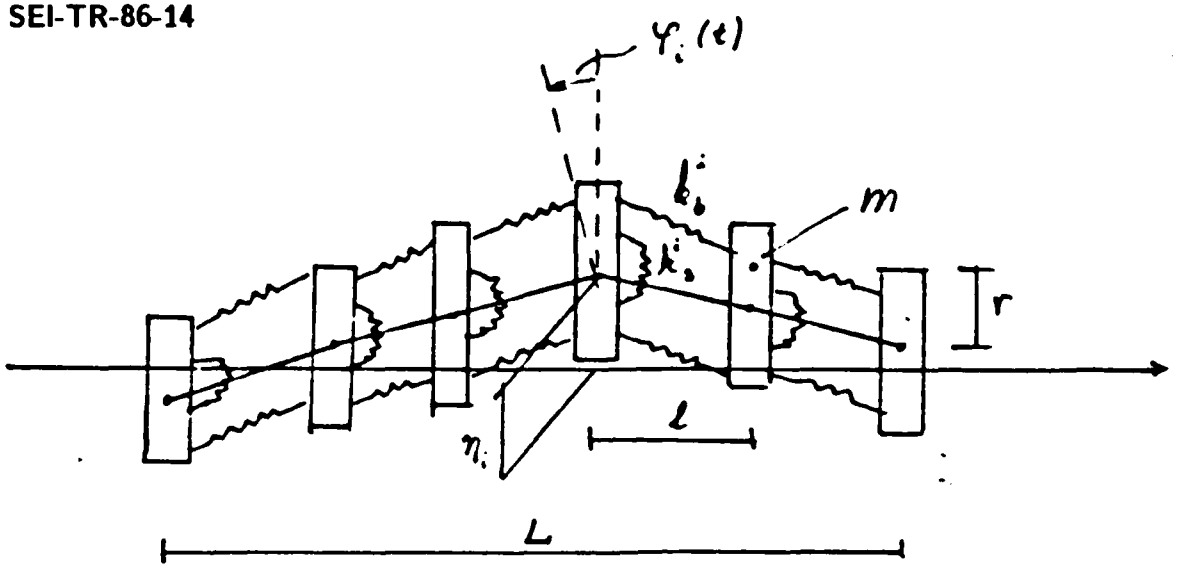


Figure 2: A lumped parameter model of the simplified truss system.

In this case the (approximate) equations of motion of the i^{th} mass are[5]

$$\frac{d^2\phi_i}{dt^2} \cong \frac{1}{r}k_a^i\{[\frac{\eta_i + 1(t) - \eta_i(t)}{\ell}] - \phi_i(t)\} \tag{161}$$

$$+S_\ell^-\{k_b^i[\frac{\phi_i + 1(t) - \phi_i(t)}{\ell}]\}$$

$$\frac{d^2\eta_i}{dt^2} = S_\ell^i\{k_a^i[\frac{\eta_i(t) - \eta_{i+1}(t)}{\ell}] - \phi_i(t)\} \tag{162}$$

where we have normalized $M = 1$ and defined

$$S_\ell^-\eta_i \stackrel{\text{def}}{=} \frac{1}{\ell}[\eta_{i-1} - \eta_i] \tag{163}$$

and similarly for $S_\ell^-\phi_i$.

To proceed, we shall introduce the nondimensional variable $\epsilon = \ell/L$ and rewrite the system (161)(162) as

$$r\frac{d^2\phi_i^\epsilon}{dt^2} = \frac{1}{r}K_a^i\{\nabla^{\epsilon+}\eta_i^\epsilon(t) - \phi_i^\epsilon\} + \nabla^{\epsilon+}\{K_b^i\nabla^{\epsilon-}\phi_i^\epsilon(t)\} \tag{164}$$

$$\frac{d^2\eta_i^\epsilon}{dt^2} = -\nabla^{\epsilon-}\{K_a^i[\nabla^{\epsilon+}\eta_i^\epsilon(t) - \phi_i^\epsilon(t)]\}$$

where

$$K_a^i = k_a^i\ell, K_b^i = k_b^i\ell \tag{165}$$

$$\nabla^{\epsilon+}\eta_i = \frac{1}{\epsilon}[\eta_{i+1} - \eta_i], \nabla^{\epsilon-}\eta_i = \frac{1}{\epsilon}[\eta_i - \eta_{i-1}].$$

[5]The spring constants depend on i since they represent the restorative forces of flexed bars, bent by different amounts.

Normalizing $\ell = 1$, we associate a position $x \in [-1/2, 1/2]$ with each mass; and we introduce the notation

$$\eta(t, x_i) = \eta_i(t), \phi(t, x_i) = \phi_i(t). \tag{166}$$

Having normalized $\ell = 1$, we have $\epsilon = \ell$ and $x_{i+1} = x_i + \ell = x_i + \epsilon$. Let $\tilde{Z} = \{x_i\}$ be the set of all points in the system. In this notation

$$(\nabla^{\epsilon+}\eta)(t,x) = \frac{1}{\epsilon}[\eta(t, x+\epsilon) - \eta(t,x)] \tag{167}$$

$$(\nabla^{\epsilon-}\eta)(t,x) = \frac{1}{\epsilon}[\eta(t,x) - \eta(t, x-\epsilon)], x \in \tilde{Z}$$

and the system is

$$\begin{aligned}\frac{d^2\phi^\epsilon(t,x_i)}{dt^2} &= K_s(x_i)\{\nabla^{\epsilon+}\eta^\epsilon(t,x_i) - \phi^\epsilon(t,x_i)\} \\ &\quad + r\nabla^{\epsilon+}\{K_b(x_i)\nabla^{\epsilon-}\phi^\epsilon(t,x_i)\} \qquad (168)\\ \frac{d^2\eta^\epsilon(t,x_i)}{dt^2} &= -\nabla^{\epsilon-}\{K_s(x_i)[\nabla^{\epsilon+}\eta^\epsilon(t,x_i) - \phi^\epsilon(t,x_i)]\}, x \in \tilde{Z}\end{aligned}$$

The scaling of (168) may be interpreted in the following way: Formally, at least, the right sides of both terms in (168) are $O(\epsilon^{-2})$. This implies that the time variations are taking place in the "fast time scale" $\tau = t/\epsilon$. Also, the spatial variations are taking place in the "microscopic scale" x which varies in ϵ-increments (e.g., $x_{i+1} = x_i + \epsilon$). Introducing the macroscopic scale $z = \epsilon x$, and the slow time scale $\sigma = \epsilon\tau$, we may rescale (168) and observe its dynamical evolution on the large space-time scale on which macroscopic events (e.g., "distributed phenomena") take place.

Rewritten in this spatial scale, the system becomes

$$\begin{aligned}\frac{d^2\phi^\epsilon(t,z_i{}^\epsilon)}{dt^2} &= \frac{1}{\epsilon}K_s(z_i{}^\epsilon)\{\delta^{\epsilon+}\eta(t,z_i{}^\epsilon) - \phi^\epsilon(t,z_i\epsilon)\} \\ &\quad + \frac{1}{\epsilon^2}\delta^{\epsilon+}\{rK_b(z_i{}^\epsilon)\delta^{\epsilon-}\phi^\epsilon(t,z_i{}^\epsilon)\} \qquad (169)\\ \frac{d^2\eta^\epsilon(t,z_i{}^\epsilon)}{dt^2} &= \frac{1}{\epsilon^2}\delta^{\epsilon-}\{K_s(z_i{}^\epsilon)[\delta^{\epsilon+}\eta(t,z_i{}^\epsilon) - \epsilon\phi^\epsilon(t,z_i{}^\epsilon)]\} \qquad (170)\end{aligned}$$

where

$$\delta^{\epsilon\pm} = \epsilon\nabla^{\epsilon\pm} = O(1) \text{ in } \epsilon. \tag{171}$$

The essential mathematical problem is to analyze the solutions $\phi^\epsilon, \eta^\epsilon$ of (169)(170) in the limit as $\epsilon \to 0$.

4.2.2 Mathematical analysis

To proceed, we shall generalize the problem (169)(170) slightly by allowing K_s and K_b to depend on z as well as z/ϵ. This permits the restoring forces in the model system to depend on the large scale shape of the structure as well as on local deformations. We

use the method of multiple scales; that is, we introduce $y = x/\epsilon$ and look for solutions of (169)(170) in the form

$$\eta^\epsilon(t) = \eta^\epsilon(t,z,y), \phi^\epsilon(t) = \phi^\epsilon(t,z,y), \tag{172}$$

and we have

$$K_s = K_s(z,y), K_b = K_b(z,y), y = \frac{z}{\epsilon}$$

On smooth functions $\psi(z,z^\epsilon)$ the operators $\delta^{\epsilon\pm}$ satisfy

$$(\delta^{\epsilon+}\psi)(z,y) = \psi(z+\epsilon, y+1) - \psi(z,y)$$

$$= \psi(z,y+1) - \psi(z,y) + \psi(z+\epsilon,y+1) - \psi(z,y+1) \tag{174}$$

$$= (S^+\psi)(z,y) + \epsilon\frac{\partial\psi}{\partial z}(z,y+1) + \frac{1}{2}\epsilon^2\frac{\partial^2\psi}{\partial z^2}(z,y+1) + O(\epsilon^3)$$

$$(\delta^{\epsilon-}\psi)(z,y) = \psi(z,y) - \psi(z-\epsilon,y-1)$$

$$= \psi(z,y) - \psi(z,y-1) + \psi(z,y-1) - \psi(z-\epsilon,y-1)$$

$$= (S^-\psi)(z,y) - \epsilon\frac{\partial\psi}{\partial z}(z,y+1) + \frac{1}{2}\epsilon^2\frac{\partial^2\psi}{\partial z^2}(z,y-1) + O(\epsilon^3) \tag{175}$$

We assume that ϕ^ϵ and η^ϵ may be represented by

$$\phi^\epsilon(t,z,y) = \phi_0(t,z) + \epsilon\phi_1(t,z,y) + \ldots \tag{176}$$

$$\eta^\epsilon(t,z,y) = \eta_0(t,z) + \epsilon\eta_1(t,z,y) + \ldots$$

and substituting (176) in (169)(170) and using (173) (174)(175), we arrive at a sequence of equations for $(\phi_0,\eta_0),(\phi_1,\eta_1),\ldots$ by equating the coefficients of like powers of ϵ.

Starting with $\epsilon^{-2},\epsilon^{-1},\epsilon^0,\ldots$, we have

$$\frac{1}{\epsilon^2}S^+[rK(z,y)S^-\phi_0(t,z)] = 0 \tag{177}$$

which is trivially true from (175) (176). The same term involving $\eta_0(t,x)$ from (170) is trivially satisfied by the assumption (176). Continuing

$$\frac{1}{\epsilon}[S^+\{rK_b(z,y)S^-\phi_1(t,z,y)\} \tag{178}$$

$$+K_s(z,y)\{S^+\eta_0(t,z) - \phi_0(t,z)\}] = 0$$

which may be solved by using the corrector $\chi_\phi(z,y)$ and taking

$$\phi_1(t,z,y) = \chi_\phi(z,y)\phi_0(t,z) \tag{179}$$

with

$$S^+\{rK_b(z,y)S^-\chi_\phi(z,y)\} = K_s(z,y) \tag{180}$$

If we regard z as a parameter in (180), then there exists a solution χ_ϕ, unique up to an additive constant, if $K_b(z,\cdot),K_s(z,\cdot)$ are periodic in y, if there exist constants A and B so that

$$0 < A \le K_b(z,y) \le B < \infty \tag{181}$$

and if the average of $K_s(z,\cdot)$ is zero

$$\frac{1}{\ell}\int_{-L/2}^{L/2} K_s(s,y)dy = 0 \tag{182}$$

Let us *assume* that (181) (182) hold, and

$$0 < A \leq K_s(z,y) \leq B < \infty. \tag{183}$$

Considering (170), the $O(\epsilon^{-1})$ term in the asymptotic expansion is

$$\frac{1}{\epsilon}[S^-\{K_s(z,y)(S^+\eta_1(t,z,y) - \phi_0(t,z))\}] = 0. \tag{184}$$

Again we introduce the corrector $\chi_\eta(z,y)$, and take η_1 in the form

$$\eta_1(t,z,y) = \chi_\eta(z,y)\phi_0(t,z) \tag{185}$$

which gives the equation for the corrector

$$S^-\{K_s(z,y)[S^+\chi_\eta(z,y) - 1]\} = 0 \tag{186}$$

or

$$S^-\{K_s(z,y)S^+\chi_\eta(z,y)\} = K_s(z,y) - K_s(z,y-1) \tag{187}$$

By hypothesis the right side in (187) is periodic in y and has zero average (182). Hence, (187) has a periodic solution, unique to an additive constant.

Continuing, the $O(\epsilon^0)$ term in (169) is

$$\begin{aligned} &S^+\{rK_b(z,y)S^-\phi 2(t,z,y)\} + K_s(z,y)[S^+\eta_1(t,z,y) - \phi_1(t,z,y)] \\ &+ K_s(z,y)\frac{\partial\eta_0}{\partial z}(t,z) + S^+\{rK_b(z,y)\frac{\partial}{\partial z}\phi_1(t,z,y)\} \\ &+S^+\{rK_b(z,y)\frac{\partial^2}{\partial z^2}\phi_0(t,z,y)\} + \frac{\partial}{\partial z}\{rK_b(z,y+1)\}\frac{\partial}{\partial z}\phi_0(t,z) \\ &+\frac{\partial^2}{\partial z^2}\{rK_b(z,y+1)\}\phi_0(t,z) - \frac{\partial^2\phi_0}{\partial t^2} = 0. \end{aligned} \tag{188}$$

This should be regarded as an equation for ϕ_2 as a function of y with *(t,z)* as parameters. In this sense the solvability condition is as before, the average of the sum of all terms on the left in (188), except the first, should be zero. We must choose ϕ_0 so that this in fact occurs; and that defines the *limiting system.*

Using the correctors (179) (185), we must have

$$\begin{aligned} \text{Average}_{(y)}\{&\frac{\partial^2\phi_0}{\partial t^2} - \frac{\partial^2\phi_0}{\partial z^2}[S^+(rK_b(z,y)) + S^+(rK_b(z,y)\chi_\phi(z,y))] \\ &- \frac{\partial\phi_0}{\partial z}[\frac{\partial}{\partial z}(rK_b(z,y+1))] - \frac{\partial\eta_0}{\partial z}K_s(z,y) \end{aligned} \tag{189}$$

$$-\phi_0[\frac{\partial^2}{\partial z^2}(rK_b(z,y+1)) + S^+(rK_b(z,y)\frac{\partial}{\partial z}\chi_\phi(z,y))$$
$$+K_s(z,y)(S^+\chi_\eta(z,y) - \chi_\phi(z,y))]\} = 0$$

Defining the functions $EI(z), G(z)$ by the associated averages in (189), the averaged equation is

$$\frac{\partial^2\phi_0}{\partial t^2} = \frac{\partial}{\partial z}(EI(z)\frac{\partial\phi_0}{\partial z}) + G(z)\frac{\partial\eta_0}{\partial z} - H(z)\phi_0 \tag{190}$$

which is the angular component of the Timoshenko beam system (Crandall et al. [27] p. 348).

Arguing in a similar fashion, we can derive the equation for the macroscopic approximation displacement of the lattice system in terms of the "equivalent" displacement $\eta_0(t,z)$ in the Timoshenko beam system

$$\frac{\partial^2\eta_0}{\partial t^2} = \frac{\partial}{\partial z}[N(z)(\frac{\partial\eta_0}{\partial z} - \phi_0(t,z))] \tag{191}$$

4.2.3 Summary

We have shown that a simplified model of the dynamics of the truss with rigid cross sectional area may be well approximated by the Timoshenko beam model in the limit as the number of cells (proportional to L/ℓ) becomes large. The continuum beam model emerges naturally in the analysis, as a consequence of the periodicity and the scaling.

To compute the approximate continuum model, one must solve (180) and (187) (numerically) for the correctors and then compute the parameters in (190) (191) by numerically averaging the quantities in (189) (and its analog for (170)) which involve the correctors and the data of the problem.

4.3 Homogenization and Stabilizing Control of Lattice Structures

In this subsection we show that the process of deriving effective "continuum" approximations to complex systems may be developed in the context of optimal control designs for those systems. This procedure is more effective than the procedure of first deriving homogeneous – continuum approximations for the structure, designing a control algorithm for the idealized model, and then adapting the algorithm to the physical model. In fact, separation of optimization and asymptotic analysis can lead to incorrect algorithms or ineffective approximations, particularly in control problems where nonlinear analysis (e.g., of the Bellman dynamic programming equation) is required.

We shall apply the combined homogenization - optimization procedure described in subsection 4.1 (based on (Bensoussan, Boccardo, and Murat [59])) to the problem of controlling the dynamics of lattice structures like the truss structure analyzed in the previous subsection. We shall only formulate a prototype problem of this type and discuss its essential features.

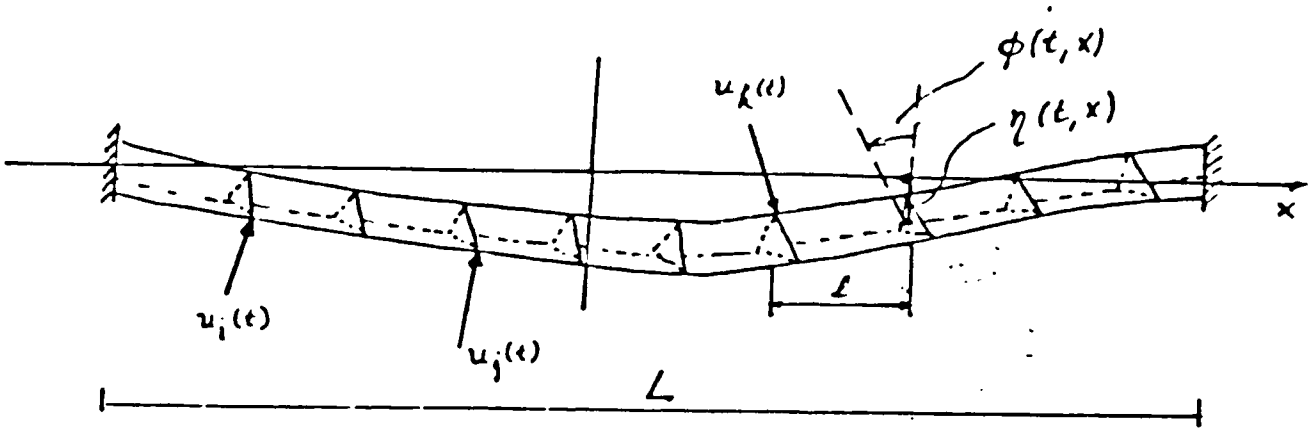


Figure 3: Truss with transverse actuator forces.

Consider the model for the lattice structure analyzed in subsection 4.3 with control actuators added. The truss shown in Figure 1 is again constrained to move in the plane and torsional motion is excluded to simplify the model and confine attention to the basic ideas. Now, however, we include a finite number of actuators acting to cause transverse motions. The truss with actuator forces indicated by arrows is shown in Figure 3. The corresponding discrete element model is shown in Figure 4.

Suppose that the physical actuators act along the local normal to the truss midline as shown in the figures, and that the forces are small so that linear approximations to transcendental functions (e.g., $sin\phi_i \cong \phi_i$, etc.) are valid. Then the controlled equations of motion of the discrete element system are (recall equation (164))

$$r\frac{d^2\phi_i^\epsilon}{dt^2} = \frac{1}{r}K_s^i\{\nabla^{\epsilon+}\eta_i^\epsilon(t) - \phi_i^\epsilon\} + \nabla^{\epsilon+}\{K_b^i\nabla^{\epsilon-}\phi_i^\epsilon(t)\} \tag{192}$$

$$\frac{d^2\eta_i^\epsilon}{dt^2} = -\nabla^{\epsilon-}\{K_s^i[\nabla^{\epsilon+}\eta_i^\epsilon(t) - \phi_i^\epsilon(t)]\} + \sum_{j=1}^{m}\delta(i,i_j)u_j(t)$$

where the notation in (165) has been used,

$$\delta(i,j) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \tag{193}$$

and $i_j, j = 1,\ldots,m$ are the locations of the actuators. Hence, if $\delta(i,i_j) = 0$ for all $j = 1,\ldots,m$ there is no actuator located at the i^{th} point which corresponds to the physical point $x \in [0,L]$. The number m of actuators is given at the outset and does not, of course, vary with the scaling. The control problem is to select the actuator forces as functions of the displacements and velocities of components of the structure to damp out motions of the structure. Measurements would typically be available from a finite number of sensors located along the structure. We shall not elaborate on this component of the model, and

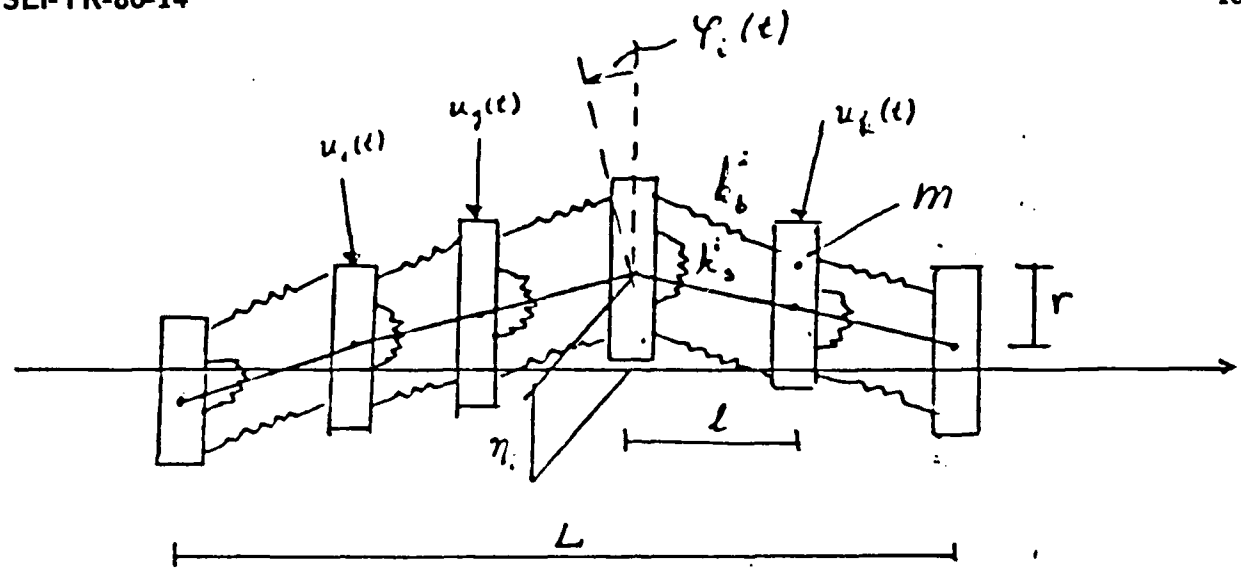


Figure 4: Discrete element model of the controlled truss.

shall instead assume that the entire state can be measured. To achieve the stabilization, we shall associate a cost functional with the system (192). Let

$$u(t) = [u_1(t), \ldots, u_m(t)]^T \tag{194}$$

be the vector of control forces, and

$$\begin{aligned} J^\gamma[u(\cdot)] = \int_0^\infty \sum_{i=1}^{N} \{ & a_i[\phi_i^\epsilon(t)]^2 + b_i[\eta_i^\epsilon(t)]^2 \\ & + \alpha_i[\dot{\phi}_i^\epsilon(t)]^2 + \beta_i[\dot{\eta}_i^\epsilon(t)]^2 \\ & + \sum_{j=1}^{m} \delta(i,i_j)u^2(t)\} e^{-\gamma t} dt \end{aligned} \tag{195}$$

where (a_i, b_i) and (α_i, β_i) are non-negative weights. Formally, the control problem is to select $\delta(i,i_j)u_j(t), i = 1, \ldots, N, j = 1, \ldots, m$ to achieve

$$\inf_{u(\cdot)} J^\gamma[u(\cdot)] \tag{196}$$

subject to (192) (193) and the appropriate boundary conditions. The case $\gamma \to 0$ corresponds to stabilization by feedback.

The analysis of this control problem is based on the scaling used in subsection 4.3, equations (164) - (171). Let $\tau = t/\epsilon$ be the fast time scale, then

$$\begin{aligned} J^\gamma[u(\cdot)] = \int_0^\infty \epsilon \sum_{i=1}^{N} \{ & a_i[\hat{\phi}_i^\epsilon(\tau)]^2 + b_i[\hat{\eta}_i^\epsilon(\tau)]^2 \\ & + \alpha_i\epsilon^2[\hat{\dot{\phi}}_i^\epsilon(\tau)]^2 + \beta_i\epsilon^2[\hat{\dot{\eta}}_i^\epsilon(\tau)]^2 \end{aligned} \tag{197}$$

$$+\sum_{j=1}^{m}\delta(i,i_j)u^2(\tau)\}e^{-\epsilon\gamma\tau}d\tau$$

with $\hat{\phi}_i^\epsilon(\tau)=\phi_i^\epsilon(\epsilon\tau)$, etc.

Let $(\phi,\dot{\phi},\eta,\dot{\eta})$ be the state vector of the system (192) with $\phi=[\phi_1,\ldots,\phi_N]^T$ and similarly for the other terms. Let $V=V^{\epsilon,\gamma}(\phi,\dot{\phi},\eta,\dot{\eta})$ be the optimal value function for the problem (192) (197). Then the Bellman equation associated with (192) (197) is

$$\epsilon\sum_{i=1}^{N}[\dot{\phi}V_{\phi_i}+\dot{\eta}_iV_{\eta_i}]$$

$$+\epsilon\sum_{i=1}^{N}\{\frac{1}{r^2}K_s^i[\nabla^{\epsilon+}\eta_i-\phi_i]+\frac{1}{r}\nabla^{\epsilon+}[K_b^i\nabla^{\epsilon-}\phi_i]\}V_{\dot{\phi}_i}$$

$$+\epsilon\sum_{i=1}^{N}\{-\nabla^{\epsilon-}[K_s^i(\nabla^{\epsilon-}\eta_i-\phi_i)]\}V_{\dot{\eta}_i} \tag{198}$$

$$\min_{u_j\in U_{ad}}\{\epsilon\sum_{i=1}^{N}\sum_{j=1}^{m}[\delta(i,i_j)u_jV_{\dot{\eta}_i}+\delta(i,i_j)u_j^2]\}$$

$$+\epsilon\sum_{i=1}^{N}[a_i\phi_i^2+b_i\eta_i^2+\epsilon^2(\alpha_i\dot{\phi}^2+\beta_i\dot{\eta}_i^2)]-\epsilon\gamma V=0.$$

Remarks:

1. Note that the minimization in (198) is well defined if the admissible range of the control forces is convex since the performance measure has been assumed to be quadratic in the control variables $\delta(i,i_j)u_j$.

2. Since we have not included the effects of noise in the model, the state equations are deterministic and the Bellman equation (198) is a first order system. To "regularize" the analysis, at least along the lines followed in conventional homogenization analysis, it is useful to include the effects of noise in the model and exploit the resulting coercivity properties in the asymptotic analysis.

3. If we introduce the macroscopic spatial scale $z=\epsilon x$, the mesh $\{x_i\}$, and the variables

$$\phi(t,z_i)=\phi_i^\epsilon(t),\dot{\phi}(t,z_i)=\dot{\phi}_i^\epsilon(t),\text{ etc.} \tag{199}$$

then the sums may be regarded as Riemann approximations to integrals over the macroscopic spatial scale z. The asymptotic analysis of (198) with this interpretation defines the mathematical problem constituting simultaneous homogenization - optimization for this case.

4.4 Effective conductivity of a periodic lattice

In this subsection we consider a version of a heat conduction problem treated by Kunnemann. Simple expressions for thermal properties of composite materials, have been derived in the past using homogenization techniques. The derivation of effective conductivities for discrete structures is useful for assessing the behavior of such structures in variable environmental conditions.

4.4.1 Problem definition

Let $Z = \{0, \pm 1, \pm 2, \ldots\}$ and $Z^d = Z \times \ldots \times Z$ (d times) be a d-dimensional lattice. Let $\epsilon > 0$ be a number small relative to 1. We want to describe the effective conduction of thermal energy on the ϵ-spaced lattice ϵZ^d. Let $e_i = (0, 0, \ldots 0, 1, 0, \ldots 0)^T$ with 1 in the i^{th} position, $i = 1,2, \ldots, d$. If x is a point in ϵZ^d, then $x \pm \epsilon e_i, 1 \leq i \leq d$, are the nearest neighbors of x. Let $a_\pm(x), x \in Z^d, 1 \leq i \leq d$, be the two functions defined on the lattice, and *assume*

$$a_i(x) \stackrel{\text{def}}{=} a_+(x) = a_{i-}(x + e_i), x \in Z^d, 1 \leq i \leq d \tag{200}$$

$$0 < A \leq a_i(x) \leq B < \infty, \forall x \in Z^d, 1 \leq i \leq d \tag{201}$$

$$a_i(x) \text{ is periodic with period } \ell \geq 1 \tag{202}$$

$$\text{in each direction, } 1 \leq i \leq d. \quad [6]$$

Next let

$$a^\epsilon_{i\pm}(x) = a_{i\pm}(\frac{x}{\epsilon}), x \in \epsilon Z^d, 1 \leq i \leq d. \tag{203}$$

Equation (201) means that the conduction process is reversible and that the conductivity $a_i(x)$ is a "bond conductivity," i.e., independent of the direction in which the bond $(x, x + e_i)$ is used by the process. Equation (203) means that the configuration of bond conductivities $a^\epsilon_{i\pm}(\cdot)$ on ϵZ^d is simply $a_{i\pm}(\cdot)$ on ϵZ^d "viewed from a distance." Assumption (202) imposes a regularity condition on the physics of the conduction process. An assumption like this is essential for existence of a limit as $\epsilon \to 0$. In one dimension the situation is illustrated in Figure 5 and Figure 6. A system similar to this with random bond conductivities was treated by Kunnemann [78] by imposing some ergodicity properties on the bond conductivities.

One can associate with this system a random (jump) process

$$\{X^\epsilon(t, x), t \geq 0, x \in \epsilon Z^d\}$$

on the ϵ-spaced lattice.[7] In effect, as $\epsilon \to 0$, $\{X^\epsilon\}$ converges to a Brownian motion on the lattice; and the main result of the analysis is an expression for the diffusion matrix $Q \stackrel{\text{def}}{=} [q_{ij}; i, j = 1, 2, \ldots, d]$ of this process. This matrix describes the macroscopic diffusion of thermal energy in the system. It is the effective conductivity.

[6] The period may be different in different directions.

[7] Definition of this process is not necessary for the analysis, but it bolsters the intuition.

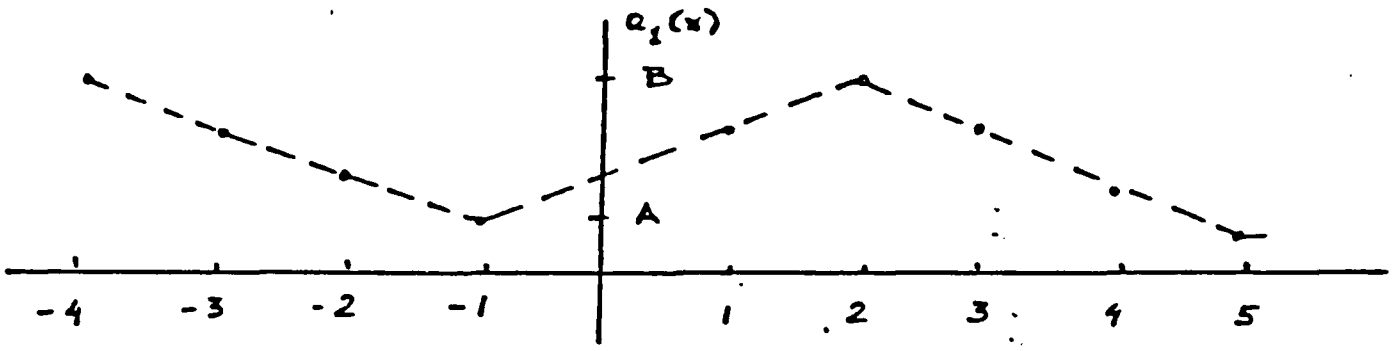


Figure 5: Conductivity on unscaled lattice with period $\ell = 6$.

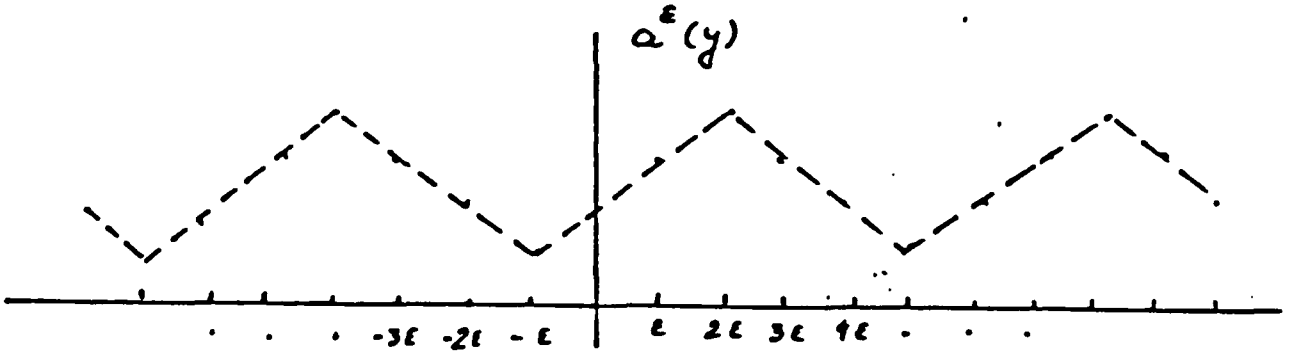


Figure 6: Conductivity on ϵ-scaled lattice, $y = \epsilon x$, $x \in Z$, period $\epsilon\ell = 6\epsilon$.

We shall carry out the asymptotic analysis of this system in the limit as $\epsilon \to 0$ using homogenization. Let

$$(\nabla_i^{\epsilon-} u)(x) \stackrel{\text{def}}{=} \frac{1}{\epsilon}[u(x - \epsilon e_i) - u(x)] \tag{204}$$

$$(\nabla_i^{\epsilon+} u)(x) \stackrel{\text{def}}{=} \frac{1}{\epsilon}[u(x + \epsilon e_i) - u(x)]$$

$$x \in \epsilon Z^d, 1 \leq i \leq d,$$

for any u square summable on ϵZ^d or square integrable on $\mathcal{R}^d$ with e_i the i^{th} natural basis vector in $\mathcal{R}^d$. Then

$$\frac{\partial u^\epsilon(t,x)}{\partial t} = -\sum_{i=1}^{d} \nabla_i^{\epsilon-}[a_i(\frac{x}{\epsilon})\nabla_i^{\epsilon+} u^\epsilon(t,x)] \tag{206}$$

$$\stackrel{\text{def}}{=} L^\epsilon u^\epsilon(t,x)$$

is the diffusion equation on the ϵ-spaced lattice with density $u^\epsilon(t,x)$ and conductivity $a_i(x/\epsilon)$. We are interested in an effective parameter representation of the thermal conduction process as $\epsilon \to 0$.

Remark: Although probabilistic methods are not required in the analysis, the associated probabilistic framework has a great deal of intuitive appeal. The operator L^ϵ may be identified as the infinitesimal generator of a pure jump process $X^\epsilon(s)$ in the "slow" time scale $s \stackrel{\text{def}}{=} \epsilon^2 t$; (Breiman [63]). Moreover, L^ϵ is selfadjoint on ϵZ^d with the inner product

$$(f,g) \stackrel{\text{def}}{=} \sum_{x \in Z^d} f(x)g(x). \tag{207}$$

Hence, the backward and forward equations for the process $X^\epsilon(s)$ are, respectively,

$$\frac{\partial p^\epsilon(y,t|x)}{\partial t} = [L^\epsilon p^\epsilon(y,t|\cdot)](x) \tag{208}$$

$$\frac{\partial p^\epsilon(y,t|x)}{\partial t} = [L^\epsilon p^\epsilon(\cdot,t|x)](y)$$

So the process is "symmetric" in the sense of Markov processes (Breiman [63]).

The asymptotic analysis of (206), when interpreted in this context, means that as the bond lattice is contracted by ϵ and time is sped up by ϵ^{-2}, the jump process $\{X^\epsilon(s)\}$ approaches a diffusion process with diffusion matrix Q. In other words, on the microscopic scale thermal energy is transmitted through the lattice by a jump process; but when viewed on a macroscopic scale the energy appears to diffuse throughout the lattice. The microscopic physics are described in (Kirkpatrick [66]) and (Kittel [67]). The approximation developed below for a periodic lattice is similar to the one developed by Kunnemann for a random lattice. This similarity demonstrates the robustness of the method, and the limted dependence of the macroscopic properties of the medium on the details of the microscopic variations of the structure.

Because the basic problem (206) is "parabolic," we can introduce the probabilistic mechanism and make use of it in the analysis. In the "hyperbolic," structural mechanical problems we treated before this device is not available.

4.4.2 Asymptotic analysis-homogenization

The essential mathematical step is to show strong convergence of the semigroup of L^ϵ, say

$$T^\epsilon(t) \stackrel{\text{def}}{=} e^{L^\epsilon t} \longrightarrow T(t) \stackrel{\text{def}}{=} e^{Lt} \text{ as } \epsilon \to 0 \tag{209}$$

and to identify the limiting operator

$$L = \sum_{i,j=1}^{d} q_{ij} \frac{\partial^2}{\partial x_i \partial x_j}. \tag{210}$$

This is accomplished by proving convergence of the resolvents

$$\text{for } \alpha > 0, [-L^\epsilon + \alpha]^{-1} \longrightarrow [-L + \alpha]^{-1} \text{ as } \epsilon \to 0 \tag{211}$$

That is, if f is a given function and

$$u^\epsilon(\cdot) \stackrel{\text{def}}{=} [-L^\epsilon + \alpha]^{-1} f \tag{212}$$

$$u(\cdot) \stackrel{\text{def}}{=} [-L + \alpha]^{-1} f$$

then $u^\epsilon \to u$ in an appropiate sense.

The method of multiple scales will be used to compute the limit. Because the conductivities $a_i(x)$ in (206) do not depend on time, we may work directly with L^ϵ rather than the parabolic PDE (206) (cf. (Bensoussan, Lions, and Papanicolaou [60]) Remark 1.6, p. 242). The method of multiple scales is convenient because it is a systematic way of arriving at the "right answers" - something which is not always simple in this analysis.

Bearing in mind (212), we consider

$$(L^\epsilon u^\epsilon)(x) = f(x) \tag{213}$$

with $u^\epsilon(x)$ in the form

$$u^\epsilon(x) = u_0(x, \frac{x}{\epsilon}) + \epsilon u_1(x, \frac{x}{\epsilon}) + \epsilon^2 u_2(x, \frac{x}{\epsilon}) + \dots \tag{214}$$

with the functions $u_j(x, y)$ periodic in $y \in \epsilon Z^d$ for every $j = 0, 1, \dots$. (As it turns out the boundary conditions are somewhat irrelevant to the construction of "right answers.") To present the computations in a simple form, it is convenient to introduce $y = x/\epsilon$, to treat x and y as independent variables, and to replace y by x/ϵ at the end.

Recall the operators $\nabla_i^{\epsilon\pm}$ from (206). Applied to a smooth function $u = u(x, x/\epsilon)$, we have

$$(\nabla_i^{\epsilon-} u)(x, y) = \frac{1}{\epsilon}[u(x - \epsilon e_i, y - e_i) - u(x, y)] \tag{215}$$

$$= \frac{1}{\epsilon}[u(x, y - e_i) - u(x, y)] + \frac{1}{\epsilon}[u(x - \epsilon e_i, y - e_i) - u(x, y - e_i)]$$

$$= \frac{1}{\epsilon}(\nabla_i^- u)(x, y) - \frac{\partial u}{\partial x_i}(x, y - e_i) + \epsilon \frac{1}{2} \frac{\partial^2 u}{\partial x_i^2}(x, y - e_i) + O(\epsilon^2)$$

where on functions $\phi = \phi(y)$

$$(\nabla_i^- \phi)(y) = \phi(y - e_i) - \phi(y) \tag{216}$$

Defining

$$(\nabla_i^+ \phi)(y) = \phi(y + e_i) - \phi(y) \tag{217}$$

we also have

$$(\nabla_i^{\epsilon +} u)(x,y) = \frac{1}{\epsilon}(\nabla_i^+ u)(x,y) + \frac{\partial u}{\partial x_i}(x, y + e_i) \tag{218}$$

$$+\epsilon \frac{1}{2}\frac{\partial^2 u}{\partial x_i^2}(x, y + e_i) + O(\epsilon^2).$$

Now we substitute (214) into (213) and use the rules (216) (217). Equating coefficients of like powers of ϵ, this leads to a sequence of equations for $u_0, u_1, \ldots$. Specifically, (using the summation convention)

$$(L^\epsilon u^\epsilon)(x,y) = -\nabla_i^{\epsilon -}[a_i(y)\nabla_i^{\epsilon +} u^\epsilon]$$

$$= \frac{1}{\epsilon^2}\nabla_i^-[a_i(y)\nabla_i^+ u_0(x,y)] - \nabla_i^{\epsilon -}[a_i(y)\frac{\partial u_0}{\partial x_i}(x, y + e_i)]$$

$$- \frac{1}{2}\epsilon \nabla_i^{\epsilon -}[a_i(y)\frac{\partial^2 u_0}{\partial x_i^2}(x, y + e_i)] + O(\epsilon) \tag{219}$$

$$-\frac{1}{\epsilon}\nabla_i^-[a_i(y)\nabla_i^+ u_1(x,y)]$$

$$-\epsilon \nabla_i^{\epsilon -}[a_i(y)\frac{\partial u_1}{\partial x_i}(x, y + e_i)] + O(\epsilon)$$

$$-\nabla_i^-[a_i(y)\nabla_i^+ u_2(x,y)] + O(\epsilon) = f(x)$$

That is, labeling each term by its order in ϵ

$$(\epsilon^{-2}) \quad \nabla_i^-[a_i(y)\nabla_i^+ u_0] = 0 \tag{220}$$

$$(\epsilon^{-1}) \quad \epsilon \nabla_i^{\epsilon -}[a_i(y)\frac{\partial u_0}{\partial x_i}(x, y + e_i)] + \nabla_i^-[a_i(y)\nabla_i^+ u_1(x,y)] = 0 \tag{221}$$

and (recall $\epsilon \nabla_i^{\epsilon \pm}$ is $O(1)$ in ϵ)

$$(\epsilon^0) \quad \frac{1}{2}\epsilon \nabla_i^{\epsilon -}[a_i(y)\frac{\partial^2 u_0}{\partial x_i^2}(x, y + e_i)] - \epsilon \nabla_i^{\epsilon -}[a_i(y)\frac{\partial u_1}{\partial x_i}(x, y + e_i)] \tag{222}$$

$$-\nabla_i^-[a_i(y)\nabla_i^+ u_2(x,y)] = f(x)$$

From (220) we have

$$a_i(y - e_i)[u_0(x,y) - u_0(x, y - e_i)] \tag{223}$$

$$-a_i(y)[u_0(x, y + e_i) - u_0(x,y)] = 0$$

If we take $u_0(x,y) = u_0(x)$, this is trivially true; and (221) simplifies to

$$\epsilon\nabla_i^{\epsilon-}[a_i(y)\frac{\partial u_0}{\partial x_i}(x)] + \nabla_i^{-}[a_i(y)\nabla_i^{+}u_1(x,y)] = 0. \tag{224}$$

At this point we introduce "correctors." That is, we assume

$$u_1(x,y) = \sum_{k=1}^{d}\chi_k(y)\frac{\partial u_0}{\partial x_k} + \hat{u}_1(x) \tag{225}$$

with $\chi_k(\cdot)$ the correctors. Using this in (224), we have (again using the summation convention)

$$\nabla_i^{-}[a_i(y)\nabla_i^{+}\chi_k(y)]\frac{\partial u_0}{\partial x_k} + [a_k(y-e_k) - a_k(y)]\frac{\partial u_0}{\partial x_k} = 0 \tag{226}$$

If we take $\chi_k(y)$ as the solution of

$$\nabla_i^{-}[a_i(y)\nabla_i^{+}\chi_k(y)] + [a_k(y-e_k) - a_k(y)] = 0 \tag{227}$$

(we have to verify the well-posedness of (227)), then (226) is satisfied. (The term $\hat{u}_1(x)$ is determined (formally) from the $O(\epsilon)$ term in the system (214) (219).)

Regarding the well-posedness of (227), note that

$$\nabla_i^{-}[a_i(y)\nabla_i^{+}\phi(y)] = \psi(y) \tag{228}$$

has a *periodic* solution on ϵZ which is unique up to an additive constant iff the average of the function $\psi(y)$ over a period $(\epsilon\ell)$ is zero; i.e.,

$$\bar{\psi} \stackrel{\text{def}}{=} \frac{1}{L}\sum_{k=1}^{L}\psi(y + ke_N) = 0, n = 1,2,\ldots,d. \tag{229}$$

This condition clearly holds in (227), and so, $\chi_k(y)$ is well defined (up to an additive constant).

We shall determine the equation for $u_0(x)$ by using (225) (227) in (222). Using the Kronecker delta function δ_{ik}, we have

$$\begin{aligned}\frac{1}{2}\epsilon\nabla_i^{\epsilon-}[a_i(y)\delta_{ik}]\frac{\partial^2 u_0}{\partial x_i\partial x_k} - \epsilon\epsilon\nabla_i^{\epsilon-}[a_i(y)\chi_k(y+e_i)]\frac{\partial^2 u_0}{\partial x_i\partial x_k} &= f(x)\\ = \{\frac{1}{2}\nabla_i^{-}[a_i(y)\delta_{ik}] - \nabla_i^{-}[a_i(y)\nabla_i^{+}\chi_k]\}\frac{\partial^2 u_0}{\partial x_i\partial x_k} &\qquad (230)\\ -\nabla_i^{-}[a_i(y)\chi_k(y)]\frac{\partial^2 u_0}{\partial x_i\partial x_k} - \nabla_i^{-}[a_i(y)\nabla_i^{+}u_2] &= f(x)\end{aligned}$$

The term in braces is zero from (227). To obtain the solvability condition (229) for u_2 in (230), we introduce the average

$$\frac{1}{2}q_{ik} \stackrel{\text{def}}{=} \text{ symmetric part } \{\overline{-\nabla_i^{-}[a_i(y)\chi_k(y)]}\} \tag{231}$$

Then solvability of (230) for u_2 gives the equation

$$\frac{1}{2}\sum_{i,k=1}^{d} q_{ik}\frac{\partial^2 u_0}{\partial x_i \partial x_k} = f(x). \tag{232}$$

And this is the diffusion equation which defines the limiting behavior of the system (213) in the macroscopic x-scale in the limit as $\epsilon \to 0$.

We can justify the asymptotic analysis by using energy estimates or probabilistic methods as in (Bensoussan, Lions, and Papanicolaou [60]). (See also Kunnemann lobal set key [78]).) We shall omit this analysis here.

4.4.3 Summary

Returning to the original problem (206) for the evolution of thermal energy on a microscopic scale, we have shown that the thermal density $u^\epsilon(t,x) \to u_0(t,x)$ as $\epsilon \to 0$ (in an appropriate norm) where

$$\frac{\partial u_0}{\partial t} = \frac{1}{2}\sum_{i,j=1}^{d} q_{ij}\frac{\partial u_0}{\partial x_i \partial x_j} \tag{233}$$

with

$$q_{ij} = -\frac{1}{L}\sum_{k=1}^{L}\{\nabla_i^+[a_i(y)\chi_k(y)] + \nabla_k^-[a_i(y)\chi_i(y)]\} \tag{234}$$

with the correctors $\chi_k, k = 1,2,\ldots,d$, given by

$$\sum_{i=1}^{d}\nabla_i^-[a_i(y)\nabla_i^+\chi_i(y)] = -[a_k(y-e_k) - a_k(y)] \tag{235}$$

$$k = 1,2,\ldots,d$$

To compute the limiting "homogenized" model (233), one must solve the system (235) (numerically) and then evaluate the average (234).

The fact that the original problem (206) is "parabolic" (i.e., it describes a jump random process), enables us to exploit the associated probabilistic structure to anticipate and structure the analysis. In this way we can anticipate that the limit problem will involve a diffusion process. In fact, the arguments used are entirely analytical and the limiting diffusion (233) is constructed in a systematic way. It is not postulated.[8]

[8]Probabilistic arguments can be used (Bensoussan, Lions, and Papanicolaou [60], Chapter 3); and they have some advantages.

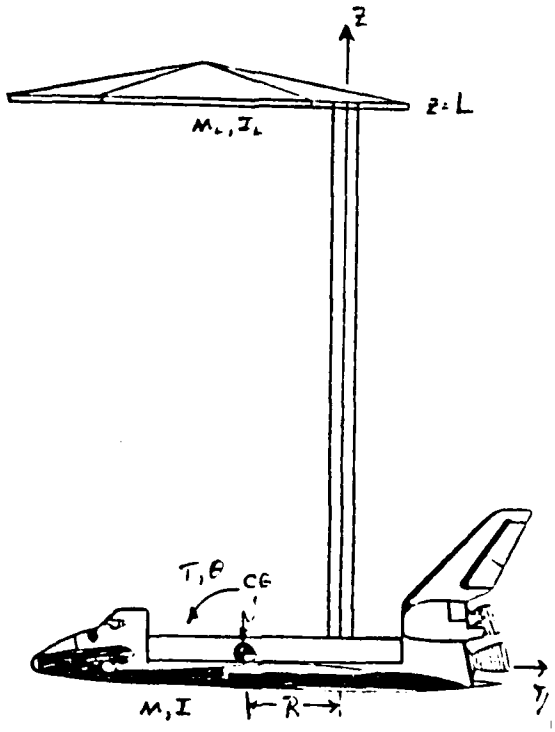


Figure 1: SCOLE Model

5 Examples

5.1 Continuum modeling for SCOLE problem

The SCOLE design challenge provides a useful baseline for comparison of continuum modeling and control design for vibration problems typically encountered in large spacecraft. The configuration is discussed in detail in [79] and consists of a large reflector antenna mounted on the end of a long truss which is attached to standard pallet in the shuttle orbiter cargo bay. The simplified model discussed in [84] includes a standard Bernoulli-Euler beam model for the flexible truss and both the space shuttle and the antenna are modeled as rigid bodies. Thus the problem can be considered as a "barbell" with flexible bar connecting the masses. In this section we provide a *linear* hybrid model for this configuration by considering motion in a plane. The configuration is shown in Figure 5.1.

The required equations of motion can be easily derived from by application of Hamilton's principle. In terms of the dynamic variables listed in Table 2 we can write the Lagranian as

$$L(\theta,\eta,\phi,\eta_L,\phi_L) = \int_0^L \left[\frac{1}{2}\rho A\left(\frac{\partial\eta}{\partial t}+(z+R)\dot{\theta}\right)^2+\frac{1}{2}\rho I\left(\frac{\partial\phi}{\partial t}+\dot{\theta}\right)^2 - \frac{1}{2}EI\left(\frac{\partial\phi}{\partial z}\right)^2-\frac{1}{2}\kappa GA\left(\frac{\partial\eta}{\partial t}-\phi\right)\right]dz \tag{236}$$

T	torque applied to shuttle
θ	angular attitude of shuttle
κG	effective shear modulus of truss
ρ	mass density of truss
A	cross sectional area of truss
E	modulus of elasticity of truss
I	moment of inertia of truss
L	truss length
M_L, I_L	mass and moment of inertia for antenna
M, I	mass and moment of inertia of shuttle (about composite CG)
R	distance from point of truss attachment to CG
z	longitudinal coordinate along truss
$\eta(z)$	lateral displacement of truss
$\phi(z)$	angular rotation of cross section of truss

Table 2: Notation for SCOLE planar motion model

$$+\frac{1}{2}I\dot{\theta}^2 + \frac{1}{2}M_L\left(\frac{\partial \eta_L}{\partial t} + (L+R)\dot{\theta}\right)^2 + \frac{1}{2}I_L\left(\frac{\partial \phi_L}{\partial t} + \dot{\theta}\right)^2.$$

Then by application of the standard variational calculus [27] to Lagrange's equation,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = Q,$$

the following equations of motion result. The Distributed Parameter Subsystem (DPS) is

$$\begin{aligned} \frac{\partial^2 \eta}{\partial t^2} &= \frac{\kappa G}{\rho}\left(\frac{\partial^2 \eta}{\partial z^2} - \frac{\partial \phi}{\partial z}\right) - (z+R)\ddot{\theta} \\ \frac{\partial^2 \phi}{\partial t^2} &= \frac{E}{\rho}\frac{\partial^2 \phi}{\partial z^2} + \frac{\kappa G A}{\rho I}\left(\frac{\partial \eta}{\partial z} - \phi\right) - \ddot{\theta} \end{aligned} \tag{237}$$

with boundary conditions

$$\begin{aligned} \eta(t,0) &= 0, \qquad & \phi(t,0) &= 0 \\ \eta(t,L) &= \eta_L(t), \qquad & \phi(t,L) &= \phi_L. \end{aligned} \tag{238}$$

The Lumped Parameter Subsystem (LPS) is

$$I\ddot{\theta} + M_L\ddot{\eta}_L + I_L\ddot{\phi}_L = \kappa G A\left(\left.\frac{\partial \eta}{\partial z}\right|_{z=0} - \left.\frac{\partial \eta}{\partial z}\right|_{z=L}\right) \tag{239}$$

$$+EI\left(\left.\frac{\partial \phi}{\partial z}\right|_{z=0} - \left.\frac{\partial \phi}{\partial z}\right|_{z=L}\right) + T(t) \tag{240}$$

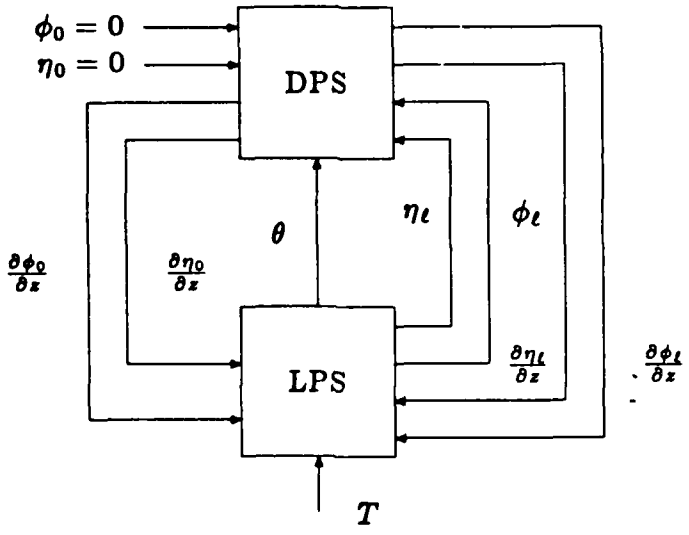


Figure 2: Subsystem interaction for SCOLE linear model

$$M_L(L+R)\ddot{\theta} + M_L\ddot{\eta}_L - \kappa GA\phi_L = -\kappa GA \left.\frac{\partial \eta}{\partial z}\right|_{z=L} \tag{241}$$

$$I_L\ddot{\theta} + I_L\ddot{\phi} = -\left.\frac{\partial \phi}{\partial z}\right|_{z=L}. \tag{242}$$

The Distributed Parameter System (DPS) interacts with the Lumped Parameter System (LPS) at the boundaries as shown in Figure 5.1. For the purposes of illustration we consider simplifying this model by assuming the shuttle mass is very much larger than that of the antenna reflector.

5.2 Cantilevered Beam Models

A natural simplification of the above model is to assume the shuttle body is so massive with respect to the antenna reflector that the vibration dynamics can be effectively studied by assuming the shuttle is rigidly fixed in space. In this section we consider the 'cantilevered' vibration model for a flexible beam with a tip mass. This is a standard problem which is derived in detail in [27, pp. 348]. For control of cantilevered vibration modes we include a control force at the antenna relector $f_u(t)$ which acts orthogonal to the longitudinal axis of the truss in it quiescent state. We state the equations of motion as:

Distributed Subsystem

$$\rho A\frac{\partial^2 \eta}{\partial t^2} = \frac{\partial}{\partial z}\left[\kappa GA\left(\frac{\partial \eta}{\partial z} - \phi\right)\right] \tag{243}$$

$$\rho I\frac{\partial^2 \phi}{\partial t^2} = \frac{\partial}{\partial z}\left[EI\frac{\partial \phi}{\partial z}\right] + \kappa GA\left(\frac{\partial \eta}{\partial z} - \phi\right) \tag{244}$$

Lumped Subsystem (at $z = L$)

$$M_L\ddot{\eta}_L + \kappa GA\left(\frac{\partial \eta}{\partial z} - \phi_L\right) = f_u(t) \tag{245}$$

$$EI\frac{\partial \phi_L}{\partial z} + I_L\ddot{\phi}_L = 0 \tag{246}$$

and with boundary conditions at $z = 0$

$$\eta(t,0) = 0 \quad \phi(t,0) = 0. \tag{247}$$

As in Section 3 we motivate the Euler-Bernoulli model by introducing the limiting argument leading to the equations

$$\rho A\frac{\partial^2 \eta}{\partial t^2} = -\frac{\partial^2}{\partial z^2}\left(EI\frac{\partial^2 \eta}{\partial z^2}\right)$$

with boundary conditions at $x = L$

$$\begin{aligned} M_L\ddot{\eta}_L &= EI\frac{\partial^3 \eta}{\partial z^3} + f_u(t), \\ EI\frac{\partial^2 \eta}{\partial z^2} &= 0 \end{aligned}$$

at $x = 0$

$$\begin{aligned} \eta(t,0) &= 0 \\ \frac{\partial \eta}{\partial z} &= 0. \end{aligned}$$

To illustrate the modeling techniques for hybrid systems and the computation of effective state space models we consider the 'long, thin beam' approximation discussed in Section 2. For this simple example we will use the approximation (discussed in Section 2 as the 'string' model) in which the Timeshenko equations are reduced by neglecting the rotation angle of cross-section $\phi = 0$ in (54). The model for the hybrid system consists of equation (69) for $0 \leq z \leq L$, subject to boundary conditions at $z = L$,

$$m_L\frac{\partial^2 \eta(t,L)}{\partial t^2} + \kappa GA\frac{\partial y(t,L)}{\partial z} = f_u(t) \tag{248}$$

and at $z = 0$,

$$\eta(t,0) = \frac{\partial \eta(t,0)}{\partial z} = 0. \tag{249}$$

Now (69) can be written in form (100)-(99) by a particular choice of distributed state: $x_d(t,z) = [\gamma, \eta]^T$ where (69) becomes with $\alpha^2 = \kappa G/\rho$

$$\begin{aligned} \frac{\partial \eta(t,z)}{\partial t} &= \alpha\frac{\partial \gamma(t,z)}{\partial z} \\ \frac{\partial \gamma(t,z)}{\partial t} &= \alpha\frac{\partial \eta(t,z)}{\partial z}. \end{aligned} \tag{250}$$

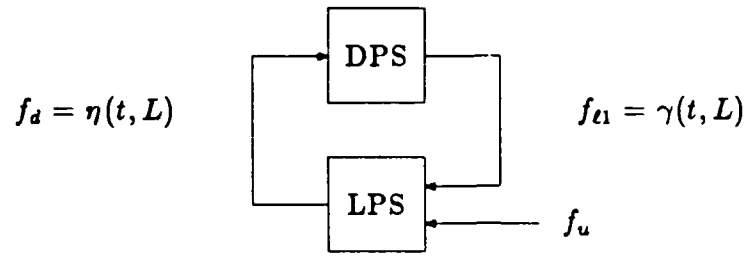


Figure 3: A Hybrid Interconnection Model

Thus we write the DPS in the canonical first-order form (100)–(99)

$$\frac{\partial x_d(t,z)}{\partial t} = \frac{\partial}{\partial z}\begin{bmatrix} 0 & \alpha \\ \alpha & 0 \end{bmatrix} x_d(t,z) \tag{251}$$

subject to

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x_d(t,0) + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} x_d(t,0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} f_d(t). \tag{252}$$

To obtain a particular state space model for the lumped system we take the LPS state as $x_\ell = [x_{\ell 1}, \eta(t,L)]^T$ where the first coordinate is chosen to satisfy $\dot{\eta}(t,L) = x_{\ell 1}(t) - \frac{\kappa GA}{\alpha}\gamma(t,L)$. The LPS model can then be written as

$$\dot{x}_\ell(t) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x_\ell(t) + \begin{bmatrix} 0 & 1/m_L \\ -\beta/\alpha m_L & 0 \end{bmatrix} \begin{bmatrix} f_{1\ell} \\ f_{2\ell} \end{bmatrix} \tag{253}$$

where $\beta = \kappa GA$. Finally, the topological interconnection is resolved by an equation of the form (105);

$$\begin{bmatrix} f_{1\ell} \\ f_{2\ell} \\ f_d \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 0 \end{bmatrix} x_d(t,L) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} x_\ell(t) = \begin{bmatrix} f_u(t) \\ 0 \\ 0 \end{bmatrix}.$$

We remark that for this example it is convenient (and relatively straightforward) to choose the state coordinates for each model so that the interactions at their boundarys are simple. This provides some insight into the meaning of the individual state variables with respect to the hybrid model as shown in Figure 5.1. From (249)–(250), we obtain the DPS Green's function of (100) using (82)–(88) which can be simplified to the following:

$$G_{r\,LEFT} = \frac{\begin{bmatrix} \sinh\frac{s}{\alpha}(w+z-L) + \sinh\frac{s}{\alpha}(w-z-L) & -\cosh\frac{s}{\alpha}(w-z-L) - \cosh\frac{s}{\alpha}(w+z-L) \\ -\cosh\frac{s}{\alpha}(w-z+L) + \cosh\frac{s}{\alpha}(w+z-L) & \sinh\frac{s}{\alpha}(w-z-L) - \sinh\frac{s}{\alpha}(w+z-L) \end{bmatrix}}{2\sinh\frac{sL}{\alpha}}$$

$$G_{r\,RIGHT} = \frac{\begin{bmatrix} \sinh\frac{s}{\alpha}(w-z+L) + \sinh\frac{s}{\alpha}(w+z-L) & -\cosh\frac{s}{\alpha}(w-z+L) + \cosh\frac{s}{\alpha}(w+z-L) \\ -\cosh\frac{s}{\alpha}(w-z+L) + \cosh\frac{s}{\alpha}(w+z-L) & \sinh\frac{s}{\alpha}(w-z+L) - \sinh\frac{s}{\alpha}(w+z-L) \end{bmatrix}}{2\sinh\frac{sL}{\alpha}}$$

with

$$G_r(s,z,w) = \begin{cases} G_{r\,LEFT}, & 0 \le z \le w \\ G_{r\,RIGHT}, & w \le z \le L \end{cases}$$

and

$$H_{BC}(s,z) = \frac{\begin{bmatrix} \cosh\frac{sz}{\alpha} \\ \sinh\frac{sz}{\alpha} \end{bmatrix}}{\sinh\frac{sL}{\alpha}}.$$

The final hybrid model can be written in terms of (106) by identifying the following terms; from (109), the hybrid Green's function G_r can be expressed using (109) given the terms

$$-H_\ell \tilde{Q}_1 P = \frac{\begin{bmatrix} -s\,\sinh\frac{sw}{\alpha} & s\,\cosh\frac{sw}{\alpha} \\ -\sinh\frac{sw}{\alpha} & \cosh\frac{sw}{\alpha} \end{bmatrix}}{\cosh\frac{sL}{\alpha} - m_L s^2 \sinh\frac{sL}{\alpha}}$$

$$-H_{BC}\tilde{Q}_2 P = \frac{\begin{bmatrix} -\sinh\frac{s}{\alpha}(w-z) - \sinh\frac{s}{\alpha}(w+z) & \cosh\frac{s}{\alpha}(w-z) + \cosh\frac{s}{\alpha}(w+z) \\ \cosh\frac{s}{\alpha}(w-z) - \cosh\frac{s}{\alpha}(w+z) & -\sinh\frac{s}{\alpha}(w-z) + \sinh\frac{s}{\alpha}(w+z) \end{bmatrix}}{2\sinh\frac{sL}{\alpha}(\cosh\frac{sL}{\alpha} - m_L s^2 \sinh\frac{sL}{\alpha})}.$$

From (108), we can write

$$\tilde{R}(s,z) = \frac{\begin{bmatrix} -m_L s\,\sinh\frac{sL}{\alpha} & -\cosh\frac{sL}{\alpha} \\ -m_L \sinh\frac{sL}{\alpha} & -m_L s\,\cosh\frac{sL}{\alpha} \\ -m_L \cosh\frac{sL}{\alpha} & -m_L s\,\cosh\frac{sL}{\alpha} \\ -m_L \sinh\frac{sL}{\alpha} & -m_L s\,\cosh\frac{sL}{\alpha} \end{bmatrix}}{\cosh\frac{sL}{\alpha} - m_L s^2 \sinh\frac{sL}{\alpha}}.$$

Finally, from (111)

$$\tilde{H}(s,z) = \frac{\begin{bmatrix} -s\,\cosh\frac{sL}{\alpha} \\ -\cosh\frac{sL}{\alpha} \\ -m_L s^2\,\cosh\frac{sz}{\alpha} \\ -m_L s^2\,\sinh\frac{sz}{\alpha} \end{bmatrix}}{\cosh\frac{sL}{\alpha} - m_L s^2 \sinh\frac{sL}{\alpha}}.$$

The above calculations were carried out using SMP. Some considerable algebraic reduction was necessary to achieve the final forms.

5.3 Control Design for Simply Supported Beam

For illustrative purpose the control design for a simply supportted ('pinned-pinned') beam is considered in some detail. Continuum modeling begins with standard Bernoulli-Euler beam equation. We develop a canonical, first-order form and identify suitable state space coordinates and identify boundary conditions which places the model in the canonical form discussed in Section 3. Computation of the required frequency response requires numerical evaluation of various transcendental terms. Numerical aspects of this are discussed.

Finally, a distributed control is computed by the method described in Section 2. We highlight the computational aspects of the procedure and, in particular, the use of a symbolic manipulation, computer algebra system to support the modeling and control design.

Testing of the distributed control law proceeds in the frequency domain. This is an advantage of the approach since frequency response data is directly available as part of the modeling and control computations. The ultimate concern for design of active control of elastic structures in space is to provide adequate stability margins in the face of the inevitable model uncertainty. The most standard methods for testing stability and assessing the robustness of control laws involve frequency domain tests. Such tests (based on Nyquist stability theory) are firmly based in engineering practice and proven reliable. The inherent robustness properties of the standard formulation for linear, quadratic optimal control which rest on frequency domain properties of the control law can be readily seen in this framework. It is essential for control of distributed parameter systems where computations will inevitably involve approximation that any degradation of the resulting stability margins be ascertainable in advance for the control configuration including sensors and actuators.

5.3.1 Continuum Modeling and State Coordinate Selection

The 'pinned-pinned' beam is shown in Figure 5.3.1. The beam is modeled by a simplified, dimensionless form of the Bernoulli-Euler equation;

$$\frac{\partial \eta^2}{\partial t^2} - 2\zeta \frac{\partial^3 \eta}{\partial t \partial z^2} + \frac{\partial^4 \eta}{\partial z^4} = 0, \tag{254}$$

where the Chen-Russell 'square-root' damping is included for material dissipation and (254) is subject to boundary conditions at $z = 0$

$$\eta(t,0) = 0 \tag{255}$$

(no lateral displacement)

$$\left.\frac{\partial^2 \eta}{\partial z^2}\right|_{z=0} = 0 \tag{256}$$

(no restraining moment) and at $z = L$ a control torque τ is applied,

$$\eta(t,L) = 0 \tag{257}$$

$$\left.\frac{\partial^2 \eta}{\partial z^2}\right|_{z=L} = \tau(t). \tag{258}$$

Writing (254) in the form

$$\frac{\partial^2 \eta}{\partial t^2} = \frac{\partial^2}{\partial z^2}\left(2\zeta \frac{\partial \eta}{\partial t} \quad \frac{\partial^2 \eta}{\partial z^2}\right)$$

identifies a new state variable γ and we write (254) in the form

$$\frac{\partial \eta}{\partial t} = \frac{\partial^2 \gamma}{\partial z^2} \tag{259}$$

$$\frac{\partial \gamma}{\partial t} = 2\zeta \frac{\partial \eta}{\partial t} \quad \frac{\partial^2 \eta}{\partial z^2} \tag{260}$$

Figure 4: Simply Supported 'pinned-pinned' Beam

Now solving for $\dot{\eta}$ in (260) and subtracting from (259) gives

$$\frac{\partial \gamma}{\partial t} = 2\zeta \frac{\partial^2 \gamma}{\partial z^2} - \frac{\partial^2 \eta}{\partial z^2}. \tag{261}$$

With the choice of state coordinates as $x = (\eta, \gamma)^T$ we can write (259) and (261) in the canonical, first-order form as

$$\dot{x} = \frac{\partial^2}{\partial z^2} \begin{pmatrix} 0 & 1 \\ -1 & 2\zeta \end{pmatrix} x. \tag{262}$$

The boundary conditions (256), (258) involve moments and can be written in terms of the state coordinates from (260) as

$$\gamma(t,0) - 2\zeta\eta(t,0) = 0 \tag{263}$$

$$\gamma(t,L) - 2\zeta\eta(t,L) = \int_0^t r(\sigma)d\sigma; \tag{264}$$

i.e., as discussed in Section 3 a generalized forcing function (torque) is evident and the integration in (264) is essential.

Finally, the equations (262)-(264) can be written in the canonical form (92), (93) identifying the matrix parameters,

$$G = \begin{pmatrix} 0 & 1 \\ -1 & 2\zeta \end{pmatrix}, \quad F = 0, \quad H = 0, \quad E = 0$$

$$\Sigma_1 = \begin{bmatrix} 1 & 0 \\ 2\zeta & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \Gamma_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 2\zeta & -1 \end{bmatrix},$$

$$\Sigma_2 = \Gamma_2 = 0, D = [0,0,0,\int \cdot dt]^T.$$

Numerical evaluation of the required frequency response data for the model in the form (81) proceeds from (95)-(99). However, direct evaluation can lead to numerical instability

depending on the required bandwidth. To see how the numercial instability arises we focus on the transfer function from the boundary control (torque at right hand end of beam) to the distributed state;

$$\hat{x}(s,z) = H_{BC}(s,z)\hat{\tau}(s),$$

which consists of two terms; viz.,

$$\begin{aligned}\hat{\eta}(s,z) &= h_{\eta\tau}(s,z)\hat{\tau}(s),\\ \hat{\gamma}(s,z) &= h_{\gamma\tau}(s,z)\hat{\tau}(s).\end{aligned}$$

An independent analysis shows that these transfer functions can be written as

$$h_{\eta\tau}(s,z) = \frac{\sinh\lambda_2 z\sin\lambda_1 L - \sin\lambda_1 z\sinh\lambda_2 L}{(\lambda_1^2+\lambda_2^2)\sin\lambda_1 L\sinh\lambda_2 L} \quad (265)$$

$$h_{\gamma\tau}(s,z) = \frac{(\zeta - i\sqrt{1-\zeta^2})\sinh\lambda_2 z\sin\lambda_1 L - (\zeta + i\sqrt{1-\zeta^2})\sin\lambda_1 z\sinh\lambda_2 L}{(\lambda_1^2+\lambda_2^2)\sin\lambda_1 L\sinh\lambda_2 L} \quad (266)$$

with

$$\begin{aligned}\lambda_1^2 &= (-\zeta + i\sqrt{1-\zeta^2})s, \quad (267)\\ \lambda_2^2 &= (\zeta + i\sqrt{1-\zeta^2})s.\end{aligned}$$

Clearly evaluation of these transfer functions at $s = 0$ by direct substitution leads to indeterminant expression $\frac{0}{0}$.

At this point a computer algebra system (SMP was used) is most useful for analysis. The system can answer the question "is there a singularity in $h_{\eta\tau}$ at $s = 0$?" For $s \to 0$ we can also let $\zeta \to 0$ which implies $\lambda_1 = \lambda_2$. Then SMP can readily perform the required power series expansion about $s = 0$ as shown. The result clearly demonstrates that the singularity is *removable*. A similar analysis for $h_{\gamma\tau}$ shows that this term has a first-order pole at $s = 0$ which confirms that the generalized torque input to this model is required for the 'new state' coordinate and is not physically evident in the lateral displacement coordinate.

Clearly, numerical evaluation of the frequency response data from this model near $s = 0$ will be numerically sensitive. We have found SMP a (somewhat) convenient tool for generation of Fortran code for numerical evaluation of these expressions. For evaluation near $s = 0$ we use SMP to generate the required power series expansions and automatically generate Fortran expression for their evaluation in 'low frequency' regions.

5.3.2 Evaluation of Frequency Response Data

In this section we demonstrate a general approach to evaluation of frequency response data for such beam models which

1. separates the integration associated with the generalized torque input to the model
2. identifies a reduced set of transcendental terms which can be analyzed independently for numerical stability and from which all frequency response data; viz., H_{BC}, G_r, can be computed.

Using the symmetry of the boundary conditions for this problem a further simplification results.

Equation (95) can be written for the 'pinned-pinned' beam as

$$\Lambda = \begin{bmatrix} 0 & I_2 \\ -sG^{-1} & \mathcal{C} \end{bmatrix} \tag{268}$$

where

$$-sG^{-1} = \begin{bmatrix} 2\zeta s & -s \\ s & 0 \end{bmatrix}.$$

As discussed in Appendix A we find it convenient for algebraic computation of the matrix exponential $\Phi(s,z) = e^{z\Lambda}$ to use a Cayley-Hamilton expansion;

$$\Phi(s,z) = \psi_0(s,z)I_4 + \psi_1(s,z)\Lambda + \psi_2(s,z)\Lambda^2 + \psi_3(s,z)\Lambda^3. \tag{269}$$

The Faddeeva algorithm (see Appendix A) is used to evaluate the scalar terms yielding,

$$\begin{aligned}
\psi_0 &= \frac{(\lambda_2^2 - 2\zeta s)\cosh\lambda_2 z + (\lambda_1^2 - 2\zeta s)\cos\lambda_1 z}{(\lambda_1^2 + \lambda_2^2)} \\
\psi_1 &= \frac{\frac{1}{\lambda_1}(\lambda_1^2 + 2\zeta s)\sin\lambda_1 z + \frac{1}{\lambda_2}(\lambda_2^2 - 2\zeta s)\sinh\lambda_2 z}{(\lambda_1^2 + \lambda_2^2)} \\
\psi_2 &= \frac{\cosh\lambda_2 z - \cos\lambda_1 z}{(\lambda_1^2 + \lambda_2^2)} \\
\psi_3 &= \frac{\frac{1}{\lambda_2}\sinh\lambda_2 z - \frac{1}{\lambda_1}\sin\lambda_1 z}{(\lambda_1^2 + \lambda_2^2)},
\end{aligned} \tag{270}$$

where λ_1, λ_2 are given in (267). These terms include all transcendental terms involved in the computation of H_{BC}, G_r. Numerical stability is an issue which can therefore be determined independently for these expressions. We also feel that this approach may suggest alternate methods for approximation of transfer functions by rational approximation. Such methods may yield efficient and numerically stable schemes for frequency response modeling.

The model equations (95)–(99) for this case can be written in a simplified form by exploiting the symmetry of the boundary conditions. From (268) we see that

$$\begin{aligned}
\Lambda^2 &= \begin{pmatrix} -sG^{-1} & 0 \\ 0 & -sG^{-1} \end{pmatrix} \\
\Lambda^3 &= \begin{pmatrix} 0 & -sG^{-1} \\ -sG^{-2} & 0 \end{pmatrix}
\end{aligned}$$

so that using (269) we can write (95) in partitioned form

$$\Phi(s,z) = \begin{bmatrix} \Phi_1 & \Phi_2 \\ \Phi_3 & \Phi_1 \end{bmatrix} \tag{271}$$

where

$$\Phi_1(s,z) = \psi_0 I_2 - s\psi_2 G^{-1} \begin{bmatrix} \psi_0(s,z) + 2\zeta s\psi_2(s,z) & -s\psi_2(s,z) \\ s\psi_2(s,z) & \psi_0(s,z) \end{bmatrix} \quad (272)$$

and

$$\Phi_2(s,z) = \psi_1 I_2 - s\psi_3 G^{-1} \begin{bmatrix} \psi_1(s,z) + 2\zeta s\psi_3(s,z) & -s\psi_3(s,z) \\ s\psi_3(s,z) & \psi_1(s,z) \end{bmatrix}. \quad (273)$$

At this point symbolic algebra system (SMP) was used to evaluate the limits of the elements in these matrices as $s \to 0$. The SMP session is reproduced as follows:

```
SMP 1.5.0
10-DEC-1986 10:56:02.05

\* the matrix phi1 is parametrized independent of s */
#I[1]::  phi1[s,z]

                2          1/2          2          1/2
              c1  Cos[c1 s    z] + c2  Cosh[c2 s    z]
#O[1]:   {{-------------------------------------------,
                          2    2
                        c1  + c2
                               1/2                 1/2
                        Cos[c1 s    z] - Cosh[c2 s    z]
                        ---------------------------------},
                                  2    2
                                c1  + c2
```

```
                                  1/2                  1/2
                    -(Cos[c1 s     z] - Cosh[c2 s     z])
                   {-----------------------------------,
                                 2     2
                              c1  + c2
                                    1/2                2
                           Cos[c1 s     z] (2zeta + c1 )
                                                  1/2                2
                                    - Cosh[c2 s     z] (2zeta - c2 )
                           ----------------------------------------}}
                                          2     2
                                       c1  + c2
#I[2]::  tmp:%;
\* rs^2 is substituted for s prior to evaluation of the limit *\
#I[3]::  tmp:S[tmp,s->rs^2]
          2                    2
        c1  Cos[c1 rs z] + c2  Cosh[c2 rs z] Cos[c1 rs z] - Cosh[c2 rs z]
#O[3]:{{-----------------------------------,--------------------------},
                          2     2                          2     2
                       c1  + c2                         c1  + c2

                    -(Cos[c1 rs z] - Cosh[c2 rs z])
                   {------------------------------,
                                 2     2
                              c1  + c2
                                                 2
                           Cos[c1 rs z] (2zeta + c1 )
                                                        2
                                    - Cosh[c2 rs z] (2zeta - c2 )
                           ---------------------------------------}}
                                          2     2
                                       c1  + c2
#I[4]::  tmp[1,1]

          2                    2
        c1  Cos[c1 rs z] + c2  Cosh[c2 rs z]
#O[4]:  -----------------------------------
                         2     2
                      c1  + c2
```

```
\* find the limit rs->0 for the 1,1 element of psi1 *\
#I[5]::  Lim[%,rs,0]

#O[5]:   1
\* find limits for each element of psi1 *\
#I[6]::  tmp[1,2]

         Cos[c1 rs z] - Cosh[c2 rs z]
#O[6]:   ----------------------------
                    2     2
                   c1  + c2

#I[7]::  Lim[%,rs,0]

#O[7]:   0

#I[8]::  tmp[2,1]

         -(Cos[c1 rs z] - Cosh[c2 rs z])
#O[8]:   ------------------------------
                     2     2
                    c1  + c2

#I[9]::  Lim[%,rs,0]

#O[9]:   0

#I[10]::  tmp[2,2]

                                  2                                  2
         Cos[c1 rs z] (2zeta + c1 ) - Cosh[c2 rs z] (2zeta - c2 )
#O[10]:  ---------------------------------------------------------
                                    2     2
                                   c1  + c2
#I[11]::  Lim[%,rs,0]

#O[11]:   1
```

```
\* perform similar limiting analysis for psi2 *\
#I[12]::  tmp:S[phi2[s,z],s->rs^2]

            2zeta (c1 Sinh[c2 rs z] - c2 Sinh[c1 rs z])

                                                 2
                                  - c1 (2zeta - c2 ) Sinh[c2 rs z]

                                                 2
                                  + c2 (2zeta + c1 ) Sin[c1 rs z]
#O[12]:   {{-------------------------------------------------------,
                                        2     2
                                c1 c2 rs (c1  + c2 )

                                Sinh[c1 rs z]   Sinh[c2 rs z]
                                ------------- - -------------
                                     c1              c2
                                -----------------------------},
                                           2     2
                                      rs (c1  + c2 )
                           Sinh[c1 rs z]   Sinh[c2 rs z]
                         -(------------- - -------------)
                               c1              c2
                        {--------------------------------,
                                    2     2
                              rs (c1  + c2 )

                                                 2
                              -(c1 (2zeta - c2 ) Sinh[c2 rs z]

                                                      2
                                        - c2 (2zeta + c1 ) Sin[c1 rs z])
                              ---------------------------------------}}
                                                 2     2
                                          c1 c2 rs (c1  + c2 )
```

```
#I[13]::  tmp[1,1]

          2zeta (c1 Sinh[c2 rs z] - c2 Sinh[c1 rs z])

                                      2
                    - c1 (2zeta - c2 ) Sinh[c2 rs z]

                                      2
                    + c2 (2zeta + c1 ) Sin[c1 rs z]
#O[13]:   ------------------------------------------
                                 2    2
                    c1 c2 rs (c1  + c2 )

#I[14]::  Lim[%,rs,0]
                              2                          2
          c1 c2 z (2zeta + c1 ) - c1 c2 z (2zeta - c2 )
#O[14]:   ---------------------------------------------
                                 2    2
                         c1 c2 (c1  + c2 )
\* factor the resulting limit to simplify *\
#I[15]::  Fac[%]

#O[15]:   z

#I[16]::  tmp[1,2]

          Sinh[c1 rs z]   Sinh[c2 rs z]
          ------------- - -------------
               c1              c2
#O[16]:   ----------------------------
                        2    2
                 rs (c1  + c2 )

#I[17]::  Lim[%,rs,0]

#O[17]:   0
```

```
#I[18]::  tmp[2,1]

                Sinh[c1 rs z]   Sinh[c2 rs z]
               -(------------- - -------------)
                     c1              c2
#0[18]:        --------------------------------
                                2     2
                         rs (c1  + c2 )

#I[19]::  Lim[%,rs,0]

#0[19]:   0

#I[20]::  tmp[2,2]

                                 2                                    2
               -(c1 (2zeta - c2 ) Sinh[c2 rs z] - c2 (2zeta + c1 ) Sin[c1 rs z])
#0[20]:        ------------------------------------------------------------------

                                                    2     2
                                          c1 c2 rs (c1  + c2 )

#I[21]::  Lim[%,rs,0]
                                  2                                  2
               -(-c1 c2 z (2zeta + c1 ) + c1 c2 z (2zeta - c2 ))
#0[21]:        -------------------------------------------------
                                       2     2
                                c1 c2 (c1  + c2 )
#I[22]::  Fac[%]

#0[22]:   z

#I[23]::  <end>
```

Now from the symmetry of the boundary conditions we can write

$$\Sigma + \Gamma\Phi = \begin{bmatrix} \Delta & 0 \\ \Delta\Phi_1 & \Delta\Phi_2 \end{bmatrix} \tag{274}$$

where

$$\Delta = \begin{pmatrix} 1 & 0 \\ 2\varsigma & -1 \end{pmatrix}.$$

It is easy to see that

$$[\Sigma + \Gamma\Phi]^{-1} = \begin{bmatrix} \Delta & 0 \\ -\Phi_2^{-1}\Phi_1\Delta & \Phi_2^{-1}\Delta \end{bmatrix}.$$

From (96) we write

$$\begin{aligned} M(s,z) &= [\Phi_1(s,z), \Phi_2(s,z)] \begin{bmatrix} \Delta & 0 \\ -\Phi_2^{-1}(s,L)\Phi_1(s,L)\Delta & \Phi_2^{-1}(s,L)\Delta \end{bmatrix} \\ &= [M_1(s,z), M_2(s,z)] \end{aligned}$$

where

$$\begin{aligned} M_1(s,z) &= \left(\Phi_1(s,z) - \Phi_2(s,z)\Phi_2^{-1}(s,L)\Phi_1(s,L)\right)\Delta \\ M_2(s,z) &= \Phi_2(s,z)\Phi_2^{-1}(s,L)\Delta. \end{aligned} \tag{275}$$

Finally, from (275) write (99) as

$$H_{BC}(s,z) = \Phi_2(s,z)\Phi_2^{-1}(s,L)\Delta \begin{bmatrix} 0 \\ 1/s \end{bmatrix}. \tag{276}$$

Substitution of (271), (275) into (99) yields

$$G_{r\,LEFT}(s;z,w) = -\Phi_2(s,z)\Phi_2^{-1}(s,L)\Phi_2(s,L-w) \tag{277}$$

$$G_{r\,RIGHT}(s;z,w) = \left[\Phi_1(s,z) - \Phi_2(s,z)\Phi_2^{-1}(s,L)\Phi_1(s,L)\right]\Phi_2(s,-w). \tag{278}$$

Numerical evaluation then proceeds by evaluating the transcendental terms (270). Substitution into (272) and (273) is followed by 2×2 matrix calculations (276)–(278). Figure 5.3.2 displays 3-dimensional frequency response data for the terms in H_{BC}. The integration in $h_{\gamma\tau}$ is evident in the response at $\omega = 0$.

5.3.3 Distributed Control Computation

The optimal control problem described in Section 1 is now completely specified by the choice of a quadratic objective which can be alternately specified by the choice of a linear operator $C : \mathcal{H}(\Omega) \rightarrow \Re^m$ as in (7). Candidates for C are:

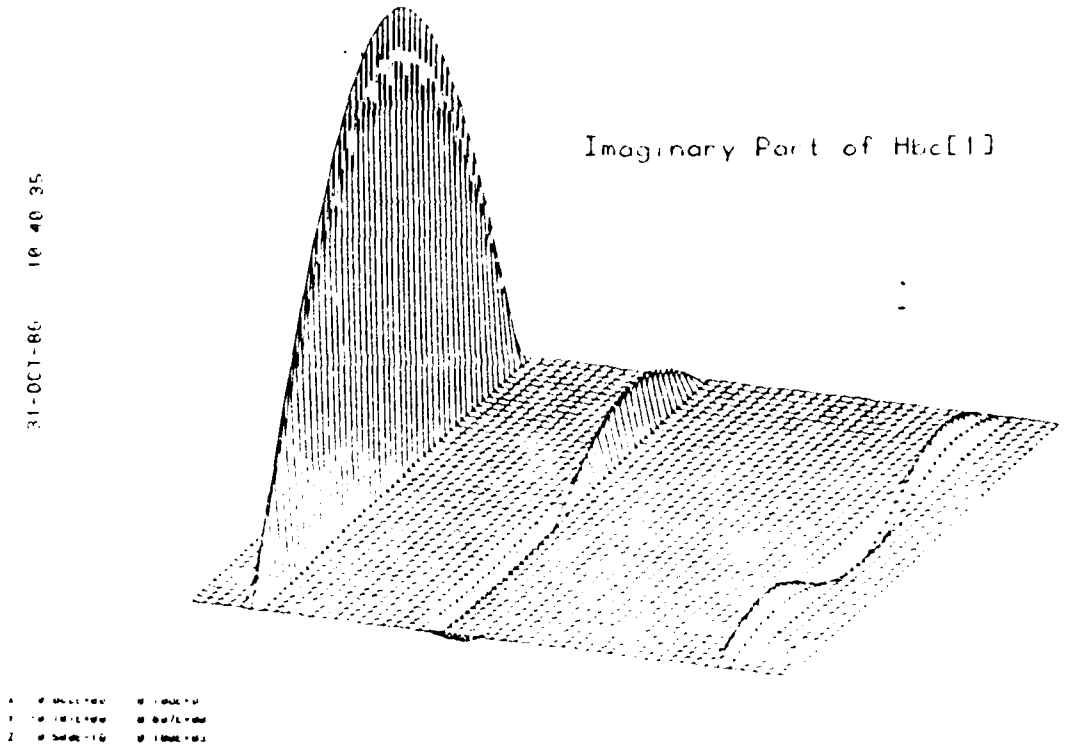
1. projection to a modal or other ROM, finite-dimensional subspace of $\mathcal{H}(\Omega)$ as discussed in Section 2
2. "point" control objective given by the operation

$$y(t) = \int_\Omega x(t,z)\delta(z - z_0)dz$$

3. weighted, state cost

$$y(t) = \int_\Omega x(t,z)w(z)dz$$

For this problem we chose a point control objective with the observation or control point at $z = L/2$. It should be noted that this choice does not suppress the fundamental difficulty of computations for distributed parameter models by employing ROM approximations. As we shall see the numerical problems are quite complex even after we have assured ourselves



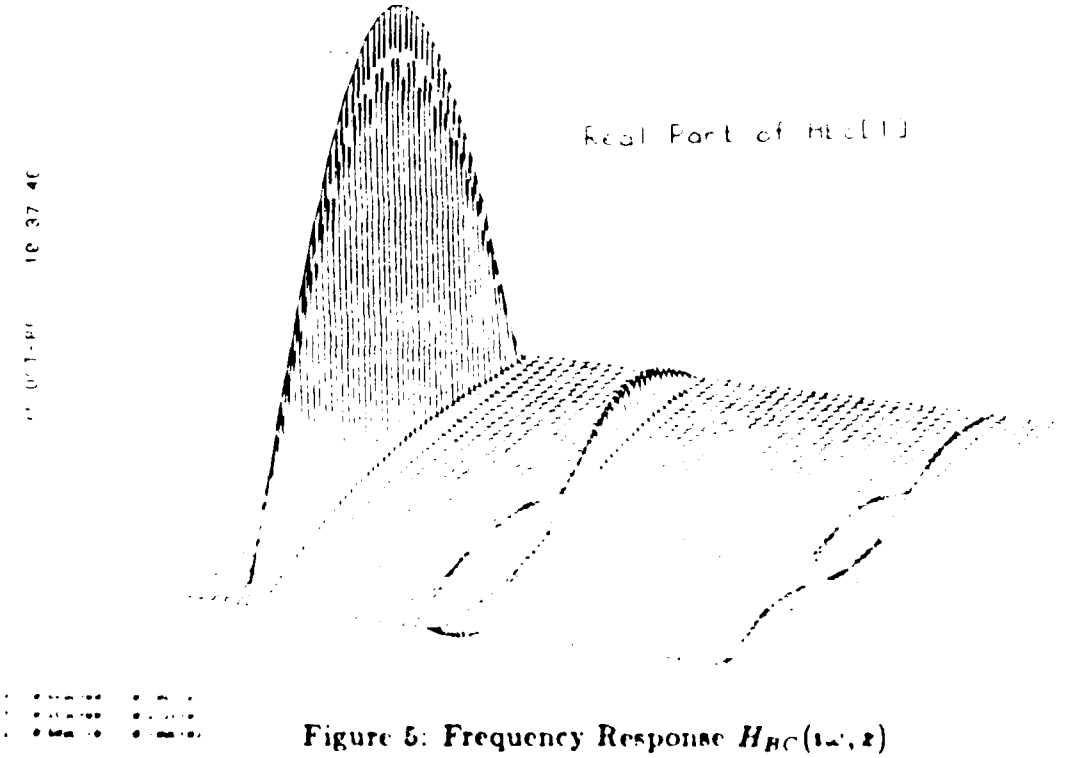


Figure 5: Frequency Response $H_{BC}(\omega, z)$

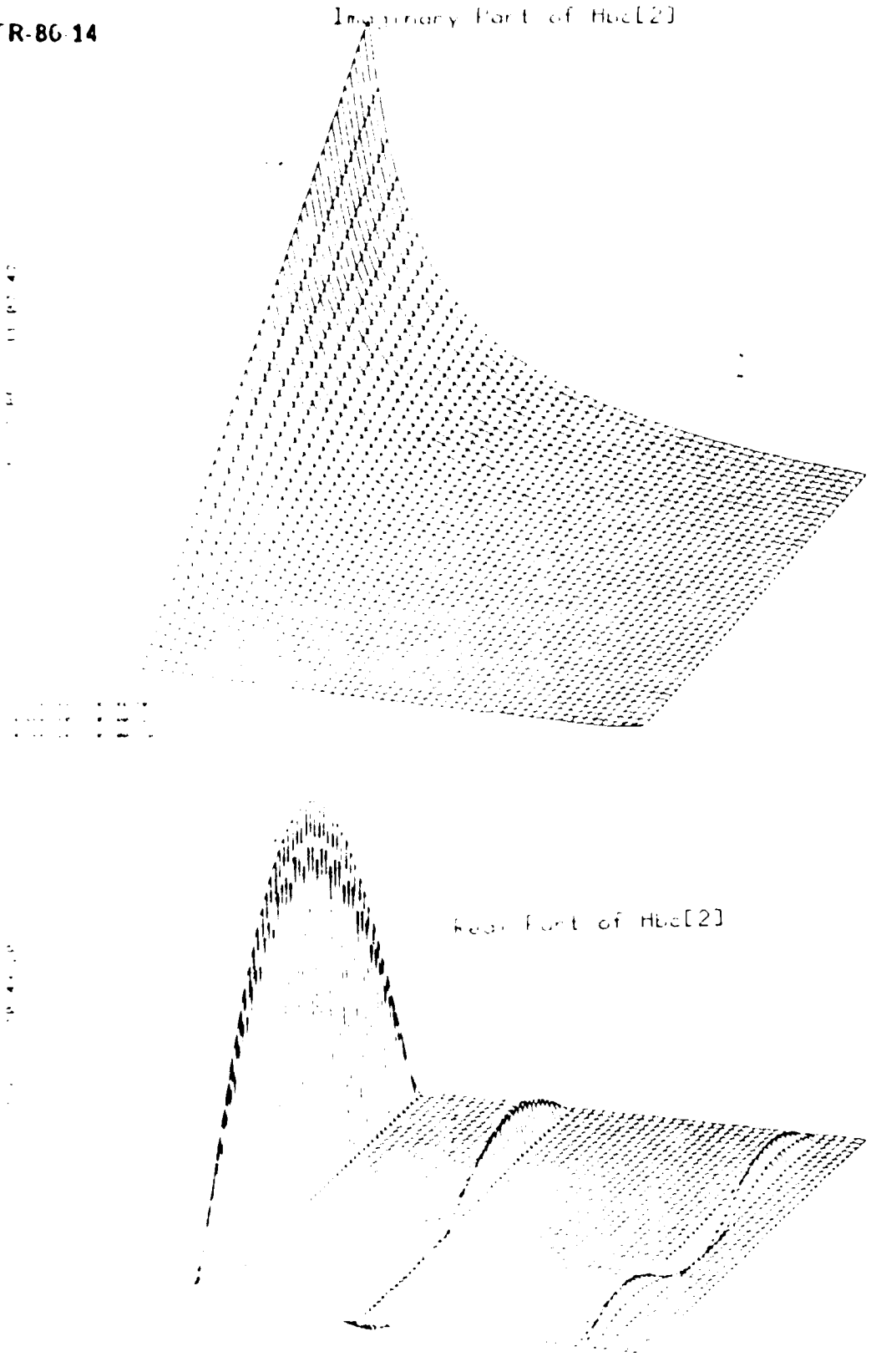


Figure 6. Frequency Response $H_{BC}(i\omega, z)$

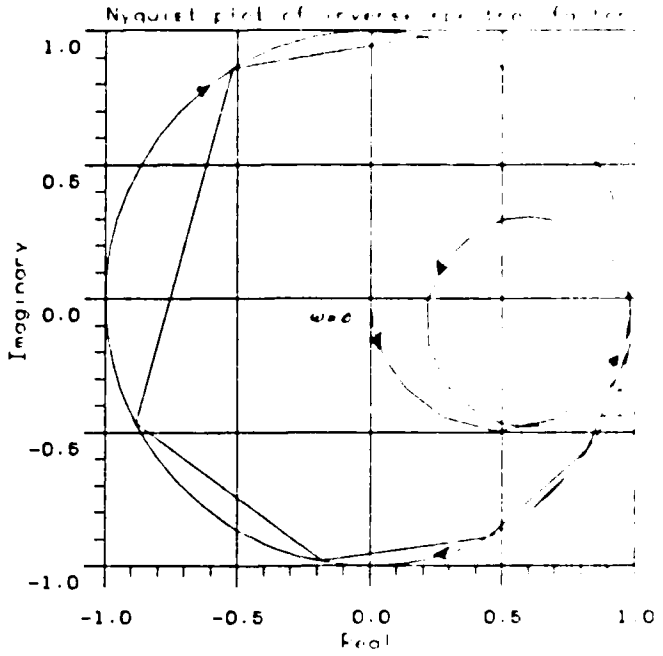


Figure 6: Nyquist plot for $1/F(i\omega)$ for beam control computation.

by various tests that the numerical evaluation of the irrational transfer functions involved can be performed with the required precision. With this choice the transfer function is

$$G(s) = H_{BC}(s, z = L/2)$$

and we must factor

$$I + G^*(i\omega)G(i\omega) = F^*(i\omega)F(i\omega)$$

to find its unique, causal spectral factor $F(i\omega)$. Using the frequency response model described above and the spectral factorization algorithm discussed in Section 2 we compute the quantity $[F^*(i\omega)]^{-1}$ which is in this case a complex scalar valued function of ω. The result is shown in Figures 5.3.3–5.3.3 with 500 samples computed over the frequency range from 1.0×10^{-10}–1.0×10^{3}.

Two observations can be made from the resulting spectral factor which help to confirm its validity. First, from basic principles of linear, quadratic control it is known [23, pp. 71] that the unique causal spectral factor is

$$F(s) = I + K_{opt}R(s; A)B$$

and enjoys the property

$$\|F(i\omega)\| \geq 1 \quad \forall \omega \in \Re.$$

Clearly from Figure 5.3.3 we see that $|F^{-1}(-i\omega)| \leq 1$ consistent with this property. Second, by choice of control observation at $z = L/2$ we see that all symmetric modes in H_{BC} do

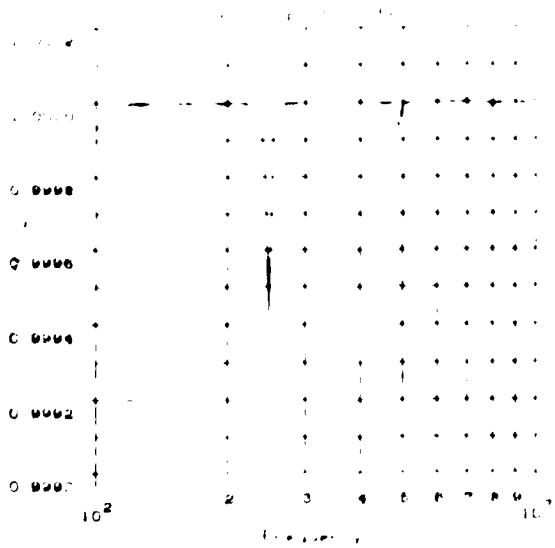


Figure 7: Gain plots for $1/F(i\omega)$ for beam control computation.

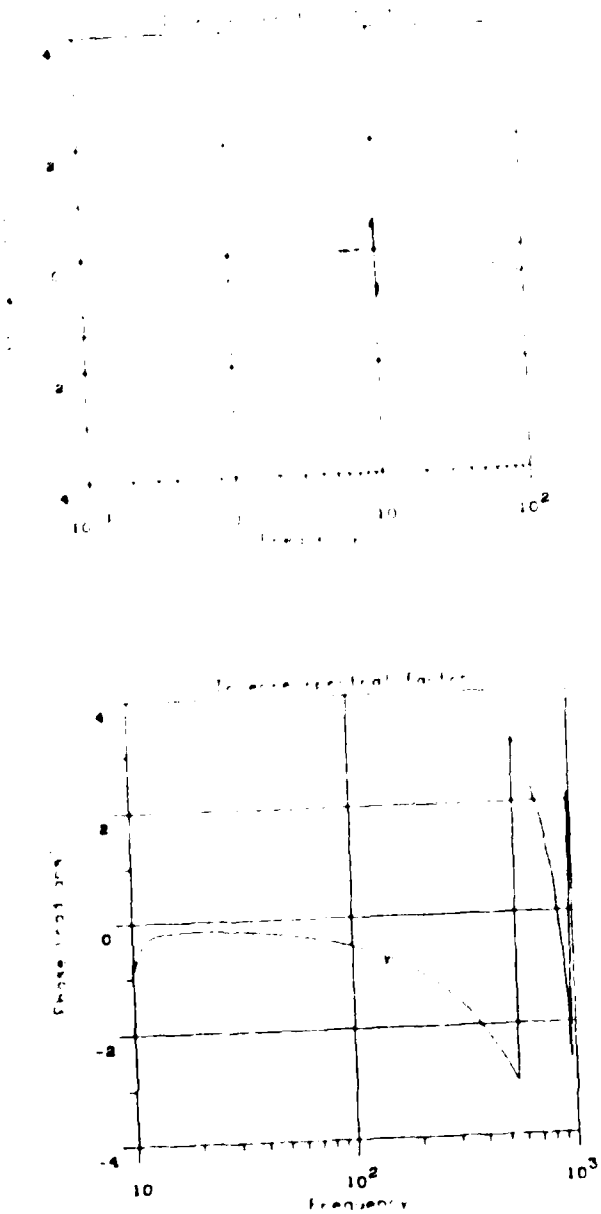


Figure 8: Phase plots for $1/F(i\omega)$ for beam control computation.

k-mode	frequency (rad/sec)
1	9.870
2	39.478
3	88.825
4	157.912
5	246.730
6	355.301
7	483.605
8	631.647
9	799.428
10	986.950

Table 3: Mode frequencies for pinned-pinned beam

not contribute to $G(s)$. From (265)-(266) it can be shown that the poles in H_{BC} occur at

$$s_k = -\varsigma k^2 L^2 \pi^2 \pm i\sqrt{1-\varsigma^2} k^2 L^2 \pi^2$$

for $k = 0, \pm 1, \pm 2, \ldots$. For these computations we have used $\varsigma = 0.005$ and $L = 1$. The first ten modal frequencies are shown in Table 2. The odd numbered (the anti-symmetric) modes only appear in the spectral factor as can be seen in the figures.

The distributed gains are computed according to (22) by numerical quadrature (trapezoidal rule). The resulting gains are displayed in Figure 5.3.3. The gains are discontinuous at the point of observation due to the discontinuity in the gradient of the Green's function. A major issue in this computation is the numerical precision obtainable by integrating over a finite bandwidth. The bandwidth used for this calculation is $\omega_0 = 1000$—the same as used for spectral factorization. Examination of the magnitude of the resulting spectral factor indicates that with the material damping model used that this bandwidth is reasonable in that the 10^{th} mode is just descernable (in double precision) in the spectral factor. Examination of the phase response indicates however that significant phase excursions are still obtained at high frequencies near modal frequencies. This phenomenon can be explained in terms of the characteristic interlacing pattern of poles and zeros for this class of transfer functions [24,30]. Because the observation point $z = L/2$ is not 'colocated' with the control point $z = L$ the transfer function will have nonminimum phase behavior. This seems to lead to poor convergence of the distributed gain computations. Figure 5.3.3 displays the convergence of two selected points of the distributed gain

$$K_{di_\bullet}(w) = \frac{1}{2\pi}\int_{-\omega_0}^{\omega_0} [F^*(i\omega)]^{-1} G^*(i\omega) C G_r(i\omega; z, w)\, d\omega$$

with bandwidth ω_0. Clearly convergence is poor as the bandwidth of the integration is increased. The effect of the rapidly varying phase is apparently significant.

The final test is to essentially 're-compute' the achieved spectral factor with the computed $K_{di_\bullet}$. This—in effect—simulates a situation where the distributed state feedback is implemented by an array of (in this case 100) sensors uniformly distributed across the

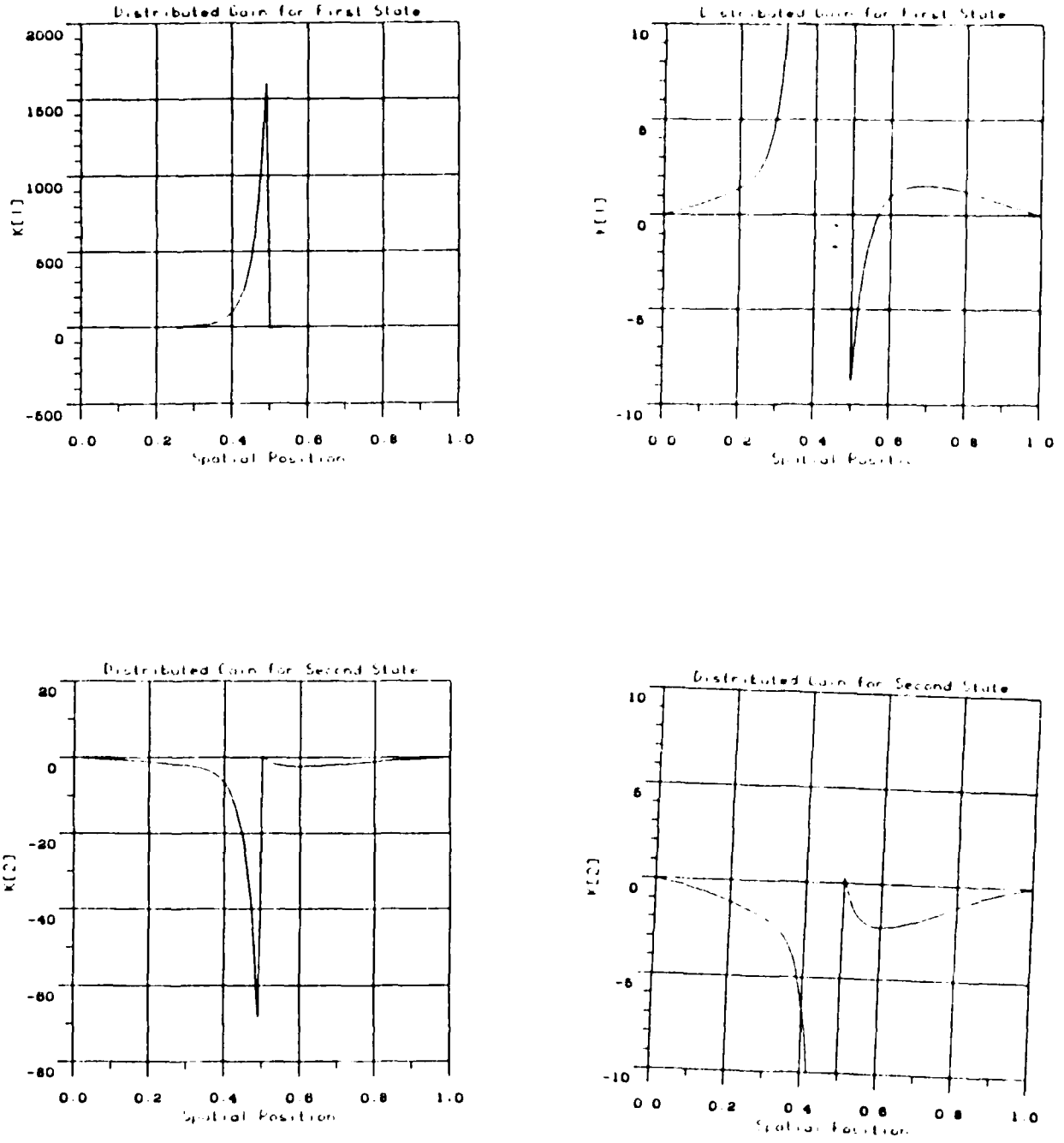


Figure 9: Distributed Gains for beam control

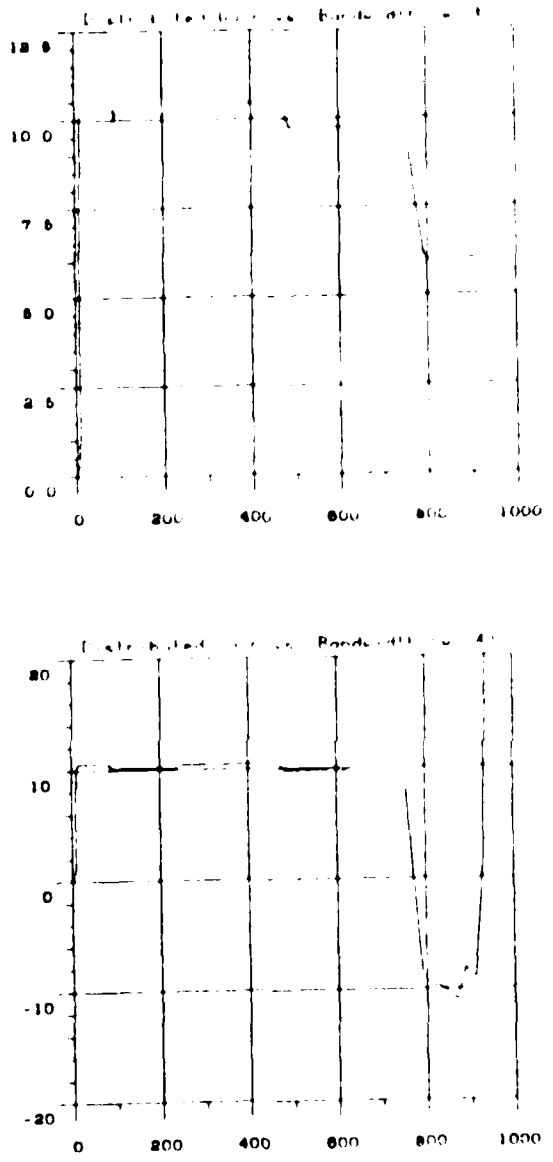


Figure 10: Convergence of two gain values with frequency

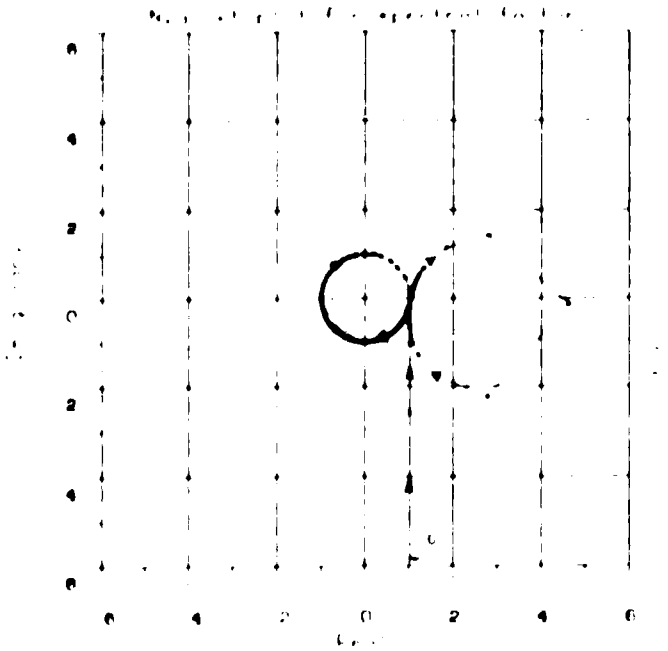


Figure 11: ideal $F(\imath\omega)$ - Nyquist plot

beam measuring both components of the state. The feedback control law is computed by numerical quadrature. It is convenient to compare the spectral factor $F(i\omega)$ computed above and displayed in Figure 5.3.3 with the 'achieved return difference' $F_a(\imath\omega) = 1 + \int_0^L K_{ss}(w)H_{BC}(\imath\omega, w)dw$ displayed in Figure 5.3.3. Figure 5.3.3 shows the characteristic behavior of the ideal spectral factor with respect to the critical point at $s = 0$. The Nyquist contour avoids the unit disk centered at the origin. In Figure 5.3.3 the achieved Nyquist contour indicates a phase error at low frequencies and for higher frequencies it approachs the critical point rather closely. The resulting control loop will be only marginally stable, will sustain a highly underdamped oscillation at some high fequency and will be extremely sensitive to modeling errors and component changes.

At this point the above negative result seems to indicate a fundamental numerical sensitivity associated with the choice of 'point' control as an optimzation objective. It seems clear that by projecting the cost onto a finite dimensional ROM subspace will lead to a much more tractable problem since the integration can now be performed over a known finite bandwidth thus assuring convergence. Of sourse as discussed in Section 2 this is equivalent to solving a finite-dimensional control problem for the ROM model so that standard methods could be applied.

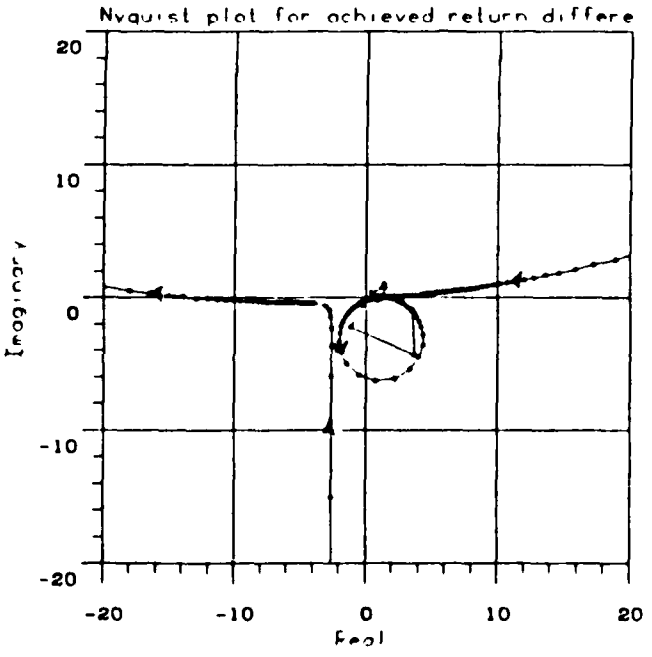


Figure 12: Achieved $F(i\omega)$ – Nyquist plot

6 Conclusions and Directions

We have considered modeling of flexible structures for the purpose of computing control laws for distributed parameter systems. We have considered both classical modeling approachs based on frequency domain analysis for well defined elastic components and homogenization for obtaining effective continuum models for periodic structures.

A complete method for computation of distributed control laws based on a standard Wiener-Hopf problem has been demonstrated. The method can, in principal, provide optimal, distributed state feedback control laws for systems which are open loop stable. In practice, numerical and other computational considerations serve to limit the effectiveness of the method to problems with well defined bandwidth and damping which provides a uniform exponential rate of decay for essentially all modes. Although methods were developed which can reliably determine appropriate irrational transfer function models for composite structures it was seen that computation of the system frequency response from such transfer functions can quickly suffer from numerical instability outside of a rather limited bandwidth.

It was shown that the proposed method embraces the standard notion of distributed control by reduced-order approximation by definition of ROM control objective. For this case the major advantage of the method appears to be the numerical approach which avoids the solution (at least explicitly) of large dimensional Riccati equation. Of course, as discussed by Davis [16], it is often desired to penalyze only those modes which are physically significant for the design at hand. In our experience is ultimately required to make such a restriction for the purposes of the numerical computations. The method of computing distributed gains is itself rather sensitive and apparently is limited in practice to computing control laws for only a finite number of modes.

The frequency response method is particularly appropriate for control system design and system performance can be ascertained in advance without recourse to simulation. By comparison of the ideal Nyquist contour for the optimal problem with various potential implimentations of distributed state sensing resulting stability and robustness properties can be seen graphically. A major difficulty with practical design of distributed state feedback is that an array of sensors spatially separated from the control points will ultimately be desired. Current advances in technology offer several opportunities for relatively low cost alternatives for such systems. However, whenever sensors are spatially separated from actuators for such systems the resulting transfer functions are nonminimum phase. Such an effect is definitely configuration dependent and results from actuator and sensor positioning but has a dominant effect on the ability to provide robust and high performance control laws.

It is significant to note the success of the spectral factorization computation for the problems considered. The algorithm is quite general and includes the case of multiple controls where the required factorization is for a matrix function. Application of this approach to the computation of output feedback control laws for distributed systems is considered in detail in [24]. Unlike the case of finite dimensional systems the output feedback design methods we have tested do not involve explicit distributed state estimation

from sensor measurements. This is a difficult problem which definitely requires ROM approximation for on-line computation. Instead we make use of alternate state realizations for a given transfer function which in many cases consist of delay-differential equations rather than partial differential equations. These equations are quite easy to include in real-time control software. Alternately, if such realizations cannot be found we can often realize the control by general purpose, real-time convolution. Such an approach is particularly significant given the increasing availablity of extremely high speed special purpose LSI chips for signal processing. Examples of these include Digital Signal Processing (DSP) chips and Fast Fourier Transform (FFT) chips.

References

[1] A.K. Noor, M.S. Anderson, & W.H. Greene, "Continuum Models for Beam- and Plate-Like Lattice Structures," *AIAA Journal*, 16, (1978) pp. 1219-1228.

[2] A. Nayfeh, M.S. Hefzy, "Continuum Modeling of Three-Dimensional Truss-Like Space Structures," *AIAA Journal*, 16, (1978) pp. 779-787.

[3] M. Anderson, "Buckling of Periodic Lattice Structures," *AIAA Journal*, 19, (1981) pp. 782-788.

[4] R. Ravelle, M. Rothstein and J. Fitch, "Computer Algebra," *Scientific American*, Dec. (1981).

[5] C. A. Cole and S. Wolfram, "SMP–A Symbolic Manipulation Program," *Proc. 1981 ACM Symp. on Sym. Alg. Comp.*, pp. 20-22 (1981).

[6] S. Wolfram, "Symbolic Mathematical Computation," *Comm. ACM*, Vol. 28, No. 4, (1985).

[7] E. Hille and R.S. Phillips, *Functional Analysis and Semigroups*, Colloquium Publications Vol 31, American Math. Soc., Providence.

[8] A. N. Michel and R. K. Miller, *Qualitative Analysis of Large Scale Dynamical Systems*, Academic Press, (1977).

[9] M. J. Balas, "Trends in Large Space Structure Control Theory: Fondest Hopes, Wildest Dreams," *IEEE Trans. Auto. Control*, AC-27,(1984), pp 522-535.

[10] J.S. Gibson, "An Analysis of Optimal Modal Regulation: Convergence and Stability", *SIAM J. Cntrl and Optim.*, (1981) Vol. 19, No. 5, pp. 686–707.

[11] M. J. Balas, "Modal Control of Certain Flexible Dynamic Systems," *SIAM J. Control*, Vol. 16, (1978) pp. 450-462.

[12] J. A. Burns and E. M. Cliff, "An Approximation Technique for the Control and Identification of Hybrid Systems," *Third VPI&SU/AIAA Symposium on Control of Large Structures*, (1981).

[13] A. S. Agarwala, *Modeling and Simulation of Hyperbolic Distributed Systems Arising in Process Dynamics*, Ph.D. Thesis, Dept. of Mechanical Eng. and Mechanics, Drexel University, Philadelphia, (June 1984).

[14] L. Meirovitch, H. Oz, "An Assessment of Methods for the Control Large Space Structures," *JACC*, Denver, Co., (1979).

[15] D. L. Russell, "Controllability and Stabilizability Theory for Linear Partial Differential Equations: Recent Progress and Open Questions," *SIAM Review*, Vol. 20, No. 4 (1978), pp 639-739.

[16] J. H. Davis and B. M. Barry, "A Distributed Model for Stress Control in Multiple Locomotive Trains," *Appl. Math. Optim.*, Vol. 3, pp. 163-190 (1977).

[17] J. H. Davis and R. G. Dickinson, "Spectral Factorization by Optimal Gain Iteration," *J. Appl. Math.*, Vol. 43, pp. 289-301 (1983).

[18] R. W. Brockett, *Finite Dimensional Linear Systems*, John Wiley, NY (1970).

[19] B. Avramovic, N. Barkakati and G. Blankenship,"Application of a Spectral Factorization Approach to the Control of Flexible Structures," *Proc. IEEE CDC*, pp. 1695-1696 (1984).

[20] B.D.O. Anderson, "An algebraic solution to the spectral factorization problem", *IEEE Trans. Auto. Cntrl.*, AC-12, pp. 410–414.

[21] J.C. Willems, "Least squares stationary optimal control and the algebraic Riccati equation," *IEEE Trans. Auto. Cntrl.*, (1971) AC-16, pp. 621–634.

[22] J.W. Helton, "A spectral factorization approach to the distributed stable regulator problem: the algebraic Riccati equation", *SIAM J. Cntrl. and Optim.*, 14, pp. 639-661.

[23] B.D.O. Anderson and J.B. Moore, *Linear Optimal Control*, (1971), Prentice Hall, Inc., Englewood Cliffs, N.J.

[24] W.H. Bennett, H.G. Kwatny, and J.S. Baras, "Frequency Domain Design of Robust Controllers for Large Space Structures," SEI-TR-86-11, Final Report NASA Contract NAS1-18209, (1986).

[25] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, (1965), McGraw-Hill Book Comp., New York.

[26] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, (1975), Prentice Hall, Inc., Englewood Cliffs, N.J.

[27] S. H. Crandall, D. C. Karnopp, E. F. Kurtz, Jr., D. C. Pridmore-Brown, *Dynamics of Mechanical and Electromechanical Systems*, McGraw-Hill Book Co., NY, (1968).

[28] I.C. Gohberg and M.G. Krein, "Systems of integral equations with kernel depending on the difference of the arguments", *Amer. Math. Soc. Trans.*, (1960) 14, pp. 217-287.

[29] C.A. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*, (1975), Academic Press, Inc., New York.

[30] B. Wie and A.E. Bryson, "Modeling and Control of Flexible Space Structures", *Proc. Dyn. Cntrl. of Lg. Flexible Spacecraft*, (1981) VPI&SU/AIAA Symposium, Blacksburg, VA., pp. 153–174.

[31] R. Piessens, E. de Doncker-Kapenga, C.W. Uberhuber, and D.K. Kahaner, *QUADPACK*, (1983), Springer-Verlag, New York

[32] F. Stenger, "The Approximate Solution of Wiener-Hopf Integral Equations", *J. Math Analysis and Applic.* (1972) 37, pp. 687-724

[33] L.R. Rabiner and R.W. Schafer, "On the Behvior of Minimax FIR Digital Hilbert Transformers", *The Bell System Tech. J.*, Vol. 53, No. 2, (Feb. 1974)

[34] L.R. Rabiner and B. Gold, *Theory and Applications of Digital Signal Processing*, (1975) Prentice-Hall, Englewood Cliffs, N.J.

[35] D.C. Youla, J.J. Bongiorno, H.A. Jabr, "Modern Wiener-Hopf Design of Optimal Controllers - Parts I and II," *IEEE Trans. on Auto. Contr.*, Vol. AC-21, pp. 3-13, and pp. 319-338

[36] M.J. Balas, "Toward a More Practical Control Theory for Distributed Parameter Systems," *Control and Dynamic Systems - Advances in Theory and Applications*, Vol. 18, C.T. Leondes, ed., 1981

[37] D.S. Bernstein and D.C. Hyland, "The Optimal Projection Equations for Finite-Dimensional Fixed-order Dynamic Compensation of Infinite-Dimensional Systems," *SIAM J. Cntrl and Optim.*, Vol. 24, No. 1, (1986), pp. 122-151

[38] P. VanDooren, "A Generalized Eigenvalue Approach for Solving Riccati Equations", *SIAM Sci. and Stat. Comp.*, Vol. 2, (1981)

[39] D.C. Youla, "On the factorization of rational matrices", *IRE Trans. Info. Theory*, (1961), IT-7, pp. 172-189

[40] R.E. Goodson, "Distributed System Simulation Using Infinite Product Expansions," *Simulation*, Vol. 15, No. 6, (1970), pp. 255-263

[41] *NASA DoD Control Structures Interaction Technology 1986*, NASA Conf. Public. 2447, Norfolk, Va., (1986).

[42] W.H. Bennett and N. Barkakati, "FlexCAD: Prototype Software for Modeling and Control of Flexible Structures," *Proc. IEEE CACSD Symp.*, Arlington, VA, (1986).

[43] Erich Zauderer, *Partial Differential Equations of Applied Mathematics*, John Wiley and Sons Inc., NY, (1983).

[44] R. W. Lyczkowski, D. Gidaspow, C. W. Solbrig and E. D. Hughes, "Characteristics and Stability Analysis of Transient One-Dimensional Two-Phase Flow Equations and Their Finite Difference Approximations," *Nuclear Science and Eng.*, 66, pp. 378-396, (1978).

[45] J. P. Sursock, "Causality Violation of Complex-Characteristic Two-Phase Flow Equations," *Intl. J. Multiphase Flow*, Vol. 8, No. 3, pp. 291-295, (1982).

46 F. John, "A Note on 'Improper' Problems in Partial Differential Equations," *Comm. Pure and Appl. Math.*, Vol. 8, pp. 591-594, (1955).

47 P. D. Lax, "On the Notion of Hyperbolicity," *Comm. Pure and Appl. Math.*, Vol. 9, pp. 267-293, (1980).

48 J. C. Firedly, *Dynamic Behavior of Processes*, Prentice-Hall, Inc., Englewood Cliffs, NJ, (1972).

49 L. Lapidus and G. F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley and Sons Inc., NY, (1982).

50 I. Stakgold, *Green's Functions and Boundary Value Problems*, John Wiley and Sons Inc., NY, (1979).

51 V. Kolousek, *Dynamics in Engineering Structures*, Butterworths, London (1973).

52 A. G. Butkovskiy, *Green's Functions and Transfer Functions Handbook*, Ellis Harwood Ltd., Chichester, England, (1982).

53 R. Piché, *Frequency-Domain Continuum Modeling and Control of Third-Generation Spacecraft*, Waterloo Res. Inst., Univ. of Waterloo, Ontario (1985).

54 G. Chen and D.L. Russell, "A Mathematical Model for Linear Elastic Systems with Structural Damping," *Quarterly J. of Applied Math.*, Vol 39, No. 4, (1982), pp. 433–454.

55 D. Poelaert, "DISTEL, a Distributed Element Program for Dynamic Modeling and Response Analysis of Flexible Structures", *Proc 4th VPI&SU/AIAA Symp. on Large Structures*, June (1983).

[56] M. Aswani 1982, *Development of an Analytical Model for Large Space Structures*, DTIC Report ADA 119349.

[57] I. Babuska 1975, *Homogenization and its applications. Mathematical and computational problems*, Technical Note BN-821, Institute for Fluid Dynamics and Applied Mathematics, University of Maryland, College Park.

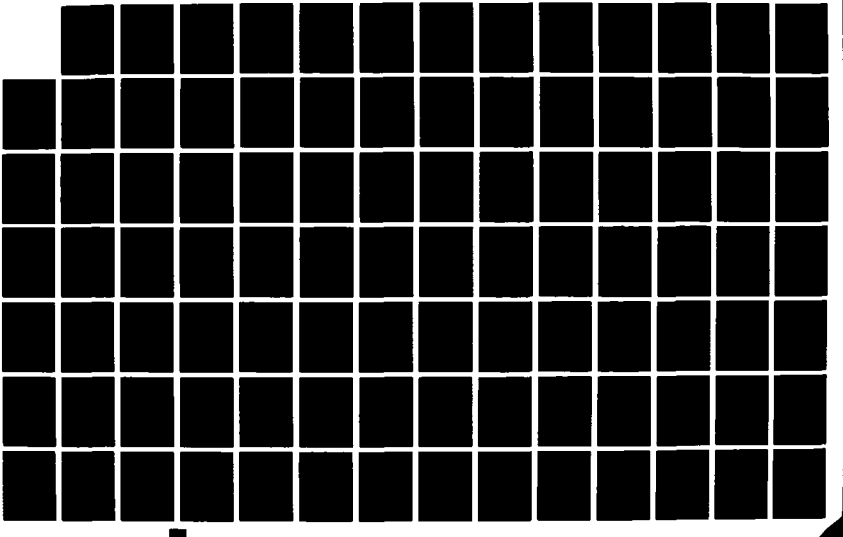
[58] A. Bensoussan 1979, "Homogenization theory," *Conf. del Seminaria de Math. dell'Univ. de Bari*, No. 158.

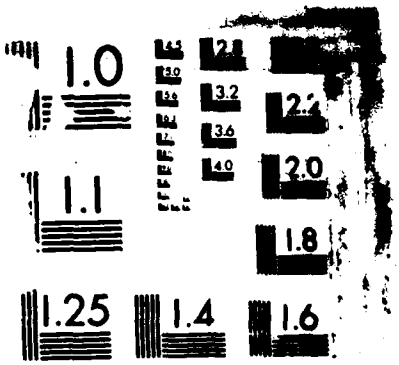
[59] A. Bensoussan, L. Boccardo, and F. Murat 1984, *Homogenization of nonlinear elliptic equations with operators not in divergence form*, preprint.

[60] A. Bensoussan, J.L. Lions, and G.C. Papanicolaou 1978, *Asymptotic Analysis for Periodic Structures*, North Holland, Amsterdam.

AD-A179 726 SPECTRAL FACTORIZATION AND HOMOGENIZATION METHODS FOR MODELING AND CONTRO.. (U) SYSTEMS ENGINEERING INC GREENBELT MD W H BENNETT ET AL. 15 DEC 86 SEI-TR-86-14 2/3

UNCLASSIFIED AFOSR-TR-87-0502 F49620-84-C-0115 F/G 22/2 NL




1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.2
2.0
1.8

[61] G.L. Blankenship 1979, "Asymptotic analysis in mathematical physics and control theory: Some problems with common features," *Richerchi di Automatica,* vol. 10, pp. 265-315.

[62] J.F. Bourgat 1978, *Numerical Experiments of the Homogenization Method for Operators with Periodic Coefficients,* INRIA Report No. 277.

[63] L. Breiman 1968, *Probability,* Addison-Wesley, Reading, Mass.

[64] D.L. Dean and S. Tauber 1959, "Solutions for one dimensional lattice structures," *J. Eng. Mech. Div., ASCE,* vol. 85, pp. 31-41.

[65] J.-N. Juang 1984, "Optimal design of a passive vibration absorber for a truss beam," *J. Guidance,* vol. 7, pp. 733-739.

[66] S. Kirkpatrick 1973, "Perculation and conduction," *Rev. Mod. Phys.,* vol. 45, pp. 574-588.

[67] C. Kittel 1976, *Introduction to Solid State Physics,* Wiley, New York.

[68] J.B. Keller 1977, "Effective behavior of heterogeneous media," in *Statistical Mechanics and Statistical Methods in Theory and Application,* U. Landman, ed., Plenum, New York, pp. 613-644.

[69] E. Larsen 1975 1976, "Neutron transport and diffusion in inhomogeneous media," I, *J. Math. Phys.,* vol. 16, pp. 1421-1427; II, *Nucl. Sci. Eng.,* vol. 60, pp. 357-368.

[70] M.M. Mikulus 1978, "Structural efficiency of long lightly loaded truss and isogrid columns for space application," NASA TM 78687.

[71] G.C. Papanicolaou and S.R.S. Varadhan 1979, "Boundary value problems with rapidly oscillating random coefficients," *Proceedings of Conference on Random Fields,* Hungary, North Holland, Amsterdam.

[72] J.D. Renton 1973, "Buckling of long, regular trusses," *Int. J. Mech. Sci.,* vol. 9, pp. 1489-1500.

[73] J.D. Renton 1970, "General properties of space grids," *Int. J. Mech. Sci.,* vol. 12, pp. 801-810.

[74] J.D. Renton 1969, "Behavior of Howe, Pratt and Warren trusses," *J. Struc. Div., ASCE.,* vol. 95, pp. 185-202.

[75] G. L. Blankenship, "Homogenization and Control of Lattice Structures," *Proc. 5th VPI&SU/AIAA Symposium on Dynamics and Control of Large Structures,* Blacksburg, VA, June (1985).

[76] G. L. Blankenship, "Homogenization of Random Lattice Structures," *Proc. CIRM Meeting Stochastic Partial Differential Equations and Applications*, Trento, (1985) to appear.

[77] J. D. Renton, "The Beam-Like Behavior of Space Trusses," *AIAA Journal*, 22, (1984), pp. 273-280.

[78] R. Kunnemann, "The Diffusion Limit for Reversible Jump Processes on Z^d with Ergodic Random Bond Conductivities," *Commun. Math. Phys.*, 90, (1983) pp. 27-68.

[79] L.W. Taylor, ed., *SCOLE Workshop Proceedings*, NASA Langley Research Center, Hampton, VA, (Dec. 1984).

[80] T. Kato,*Perturbation Theory for Linear Operators*, Springer-Verlag, New York.

[81] R.F. Curtain and A.J. Pritchard, "The infinite-dimensional Riccati equation for systems defined by evolution equations", *SIAM J. Cntrl. and Optim.*, (1976), 14, pp. 951–983.

[82] D.L. Russell, "On Mathematical Models for the Elastic Beam with Frequency-Proportional Damping," to appear.

[83] R.F. Curtain and A.J. Pritchard, "The infinite-dimensional Riccati equation for systems defined by evolution equations", *SIAM J. Cntrl. and Optim.*, (1976), 14, pp. 951–983.

[84] A.V. Balakrishnan, "A Mathematical Formulation of the SCOLE Control Problem: Part I", NASA contractor report 172581, (1985).

[85] T. Kailath, *Linear Systems*, Prentice-Hall, (1980).

A Algorithms and Code for Symbolic Manipulation

A central goal of this project was to investigate the use of a symbolic manipulation (computer algebra) language to support the required model building computations. We used the language SMP throughout this project. The goal was to automate—as much as possible—computation of the required frequency response models and to setup the equations in a form suitable numerical evaluation. Finally, the language should support the automatic generation of Fortran code which can then be linked with the required procedures for spectral factorization and optimal gain computation. The main feature of the algebraic approach is to evaluate the expressions symbolically and therefore carry through certain model parameters. This feature is quite useful for generic model building but there is a penalty in that complexity of the resulting expressions can limit their utility while limiting the computational speed and effectiveness of the computer algebra system.

The first step was to build a system shell in SMP with menu-driven, user interface which can aid the novice user. The more experienced user could then access the required SMP procedures directly. As with any computer algebra system proficiency comes only with practice and many seemingly straightforward computations can lead to undesirably large expressions. Simplification is often tedious requiring not only skill with the special algebraic relations required but also the language syntax and operations for symbolic identification of parts of expressions. The system shell we implemented included the equations for computation of the boundary transfer function and Green's function for both the hyperbolic and parabolic forms developed in Section 3. We also included the equations for computation of hybrid coordinate models.

An essential computation for these equations is evaluation of a matrix exponential. We found it convenient to use an algebraic relation based on the Laplace transform and the Cayley-Hamilton theorem for matrices. The relation is commonly credited to Leverrier-Souriau-Faddeeva-Frame [85, pp 658]. The formulae are developed as follows. Given an $n \times n$ matrix A the characteristic equation is

$$\begin{aligned} \pi_A(s) &= \det[sI - A] \\ &= a_0 + a_1 s + \ldots + a_{n-1} s^{n-1} + s^n. \end{aligned}$$

Then the Cayley-Hamilton theorem says that

$$\pi_A(A) = 0.$$

The matrix exponential

$$e^{At} = \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}$$

can then be written as a finite sum;

$$e^{At} = \sum_{k=0}^{n} \psi_k(t) A^{k-1}.$$

The coefficients can be computed as

$$\psi_k(t) = \frac{1}{2\pi i} \oint_\Gamma \frac{\pi^{(k)}(s)}{\pi_A(s)} e^{ts} ds$$

for Γ a smooth, closed contour enclosing the spectrum of A and where the polynomials $\pi^{(k)}(s)$ are generated recursively according to

$$\pi^{(j-1)}(s) = s\pi^(j)(s) + a_{j-1}\pi^{(n)}(s)$$

with

$$\pi^{(n)}(s) = 1.$$

The contour integral is the usual Laplace transform inverse which can be evaluated by residues. Most available computer algebra systems (e.g. MACSYMA, SMP, REDUCE) can perform the calculation by explicit partial fraction expansion of the rational integrand in combination with certain tables for standard inverses. This approach was programmed in SMP and tested. It proved considerably faster than the direct approach; i.e., compute e^{At} by inverse Laplace transform of the $n \times n$ matrix $[sI - A]^{-1}$.

SMP offers the facility for 'tuning' automatic simplification of expressions by adding 'rules' which are automatically applied at each computation. We investigated this feature for automatic simplication of the transcendental terms generated for transfer function computation. Unfortunately it proved to be difficult to determine a set of algebraic rules which would always lead to "desirable" reduced forms. An alternate approach we found useful was to build special simplification procedures in SMP for various steps in the computation. These procedures could be applied manually to resulting expressions. They could also be applied with the 'MAP[]' operation of SMP which recursively applies a set of procedures until the expression does not change. This can lead to indefinite recursions for certain expressions. In conclusion, we found that SMP offered several features which were useful for expression simplification but no automatic procedure could be developed which would always lead to a 'nice enough' expression that the knowledgable human could not ask for more. In the final analysis computer algebra is a tool for experts—*not* (in the current jargon) an expert system!

One aspect of computer algebra that we found most useful on this project however is the use of standard expansions for generating alternate expressions for numerically evaluating the required irrational transfer functions. This was an essential task and the understanding of the numerical stability properties of transcendental terms absorbed a major portion of this project. In order to make a computer system ultimately useful to engineers it will be necessary to include specific provisions for numerical testing and analysis. Coupling this with a feature for automatic Fortran or C code generation should provide a useful tool for advanced model development and analysis of mechanical systems. The feature for automatic Fortran code generation was not fully debugged in the version of SMP that we used. We were able to compensate for most of the bugs with a programmable editor (Gnu Emacs).

The following code is the SMP system shell we developed for this project. It includes a menu-driven, user interface containing all the required equations for computing the transfer function models for hyperbolic, parabolic, and hybrid configurations.

Menu Driven Programs

- Main Menu
- Distributed Parameter System (DPS)
- Lumped Parameter System (LPS)
- Hybrid System
- Parabolic System
- Incidence Relations
- Cost Definition

```
                /** Green's Function Menu Kernel **/

/*K:   application menu; Green's function; transfer function */

/*A: Ajaipal S. Virdy */
/*S: Systems Engineering Inc */
/*D: February 20, 1986 */


/*

    This Menu driven program computes the Green's Function for a Distributed
Parameter System, the Resolvent and transfer function for a Lumped Parameter
System, and the Green's Function for the Hybrid System.  For the Distributed
Parameter System, the state space model is
                 .
           Xd(t,z) = Ad Xd'(t,z) + Bd Xd(t,z)
                                      ' '
           SIGMA Xd(t,0) + GAMMA Xd(t,L) = Emat Fd(t)

where Ad, Bd, SIGMA, and GAMMA are Nd x Nd matrices and Emat is an Nd x Md
matrix.
   The frequency response model for the above model is

                  /L
    .             |                                     ^
   Xd(s,z) =      | Gr(s,z,w) Xd(w)dw + Hbc(s,z) Fd(s)
                  |
                 /0


   ComputeDPS computes Hbc(s,z) and the left and right Green's Function
GreenL and GreenR, respectively.  An intermediate step in the computation
of these transfer functions is the computation of Msz which is defined as:

                       -1
                    Ad   [ sI - Bd ] z
          M(s,z) =  e

(Notice the transition matrix computation involved here).  All results
depend upon the computation of this matrix exponential.  At present, the
algorithm for computing the matrix exponential works only for a second
order state space model.  Higher order models will not work with the present
algorithm because of the parameters s and z.


    The Lumped Parameter System model looks like:
         .
         Xl(t) = Al Xl(t) + Bl Fl

where Al is an Nl x Nl matrix and Bl is an Nl x Ml matrix.  ComputeLPS returns
the Resolvent and Transfer function for this model.

    The overall frequency response model for the the Hybrid System is

                  /L
      ^          ( ~            0         ~        0  ~      ~
      X(s,z) =   )  Gr(s,z,w) Xd(w)dw + R(s,z) Xl + H(s,z) U(s)
                 )
                /0
```

```
where
                 |    |
                 | Xl |
       X(s,z) =  |    |
                 | Xd |
                 |    |

ComputeHYBRID returns R(s,z), H(s,z), and the left and right half
of the Hybrid Green's Function GRszwL and GRszwR.

   FORTRAN code can be generated for any of the transfer functions computed
and the code can then be linked with MATLAB for plotting or displaying the
Bode and Nyquist plots for these transfer functions.

                                                                    */

/*  ------------------------------------------------------------------------  */

/*        S O U R C E   C O D E  F O R  G R E E N M E N U                    */

/* ------------------------------------------------------------------------- */

/*   GLOBAL SYMBOLS: variables and routine names                             */

MainMenuNum[1] ::  GreenIntroduction
MainMenuNum[2] ::  ComputeDPS
MainMenuNum[3] ::  ComputeLPS
MainMenuNum[4] ::  GetIncidenceRelation
MainMenuNum[5] ::  ComputeHYBRID
MainMenuNum[6] ::  CoPrimeResult
MainMenuNum[7] ::  AutoSimplification
MainMenuNum[8] ::  NoAutoSimplification
MainMenuNum[9] ::  CostDefinition
MainMenuNum[10] :: ComputeHs
MainMenuNum[11] :: ComputeParabolicSys
MainMenuNum[12] :: Null

MainMenuNum[88] :: ROUTINEFORHELP

/* ------------------------------------------------------------------------- */
/*                          The Main Menu                                    */
/*                         ***************                                   */
/* ------------------------------------------------------------------------- */

/*:  MainMenu
       Display the Main Menu for computation of GREEN's FUNCTION   */

MainMenu::(\
ClearScreen;\
Pr["
       * * * * * * *  M A I N   M E N U  * * * * * * *

           * * *  G R E E N s  F U N C T I O N  * * *
```

```
    1  INTRODUCTION                          2  ComputeDPS
    3  ComputeLPS                            4  GetIncidenceRelation
    5  ComputeHYBRID                         6  CoPrimeResult
    7  AutoSimplification                    8  NoAutoSimplification
    9  CostDefinition                        10 Compute H(s) = I + G*G
    11 Compute Parabolic System              12 PowerSeriesExpansion
    88 Help                                  99 Exit to SMP
"])

/* ------------------------------------------------------------------------ */

/* Initialize SMP session */

/* ClearScreen
       Clears screen by printing escape sequence  */

ClearScreen::Dsp["SEI$USR:[LSS.SMP.TOOLS]CLEAR_SCREEN.SMP"]

/*  ------------------------------------------------------------------------  */

!@INIT

Initialize::(\
Open[{,,{80,999999}}]; /*  take control of the screen  */\
If["P[_XLoadonce[Loaded]=1],<XLoadonce]; /* Load external file */\
\
   Loadonce[XPrtable];\
   Loadonce[XMat1];\
   Loadonce[XMat3]\
)

/*  ------------------------------------------------------------------------  */

/*: ComputeDPS
       Load and display menu for computing Distributed Parameter System  */

ComputeDPS::(If["P[_DPSMenu[Loaded]=1], /* Then */\
                (Pr["Loading Distributed Parameter System Menu . . ."];\
                 <"SEI$USR:[LSS.SMP]DPS_MENU.SMP")\
               ];\
             DPSMenu)

/*  ------------------------------------------------------------------------  */

/*: ComputeParabolicSys
       Load and display menu for computing Distributed Parabolic System  */

ComputeParabolicSys::(If["P[_PARAMenu[Loaded]=1], /* Then */\
                (Pr["Loading Distributed Parabolic System Menu . . ."];\
```

```
                <"SEI$USR:[LSS.SMP]PARAB_MENU.SMP")\
              ];\
            PARAMenu)

/*  ------------------------------------------------------------------------  */

/*: CostDefinition
        Load and display menu for computing the Cost Definition  */

CostDefinition::(If[~P[_CostDefMenu[Loaded]=1], /* Then */\
                 (Pr["Loading Cost Definition Menu . . ."];\
                  <"SEI$USR:[LSS.SMP]COST_DEF_MENU.SMP")\
               ];\
             CostDefMenu)

/*  ------------------------------------------------------------------------  */

/*: ComputeLPS
        Load and display menu for computing the Lumped Parameter System  */

ComputeLPS::(If[~P[_LPSMenu[Loaded]=1], /* Then */\
                 (Pr["Loading Lumped Parameter System Menu . . ."];\
                  <"SEI$USR:[LSS.SMP]LPS_MENU.SMP")\
               ];\
             LPSMenu)

/*  ------------------------------------------------------------------------  */

/*: GetIncidenceRelation
        Obtain the Incidence Relations for computing the HYBRID System  */

GetIncidenceRelation::(If[~P[_IncidenceRel[Loaded]=1], /* Then */\
                          (Pr["Loading file . . ."];\
                           <"SEI$USR:[LSS.SMP]INCID_RELA.SMP")\
                         ];\
                       GetIncidRela)

/*  ------------------------------------------------------------------------  */

/*: ComputeHYBRID
        Load and display menu for computing the HYBRID System  */

ComputeHYBRID::(If[~P[_HYBRIDMenu[Loaded]=1], /* Then */\
                   (Pr["Loading Hybrid System Menu . . ."];\
                    <"SEI$USR:[LSS.SMP]HYBRID_MENU.SMP")\
                 ];\
               HYBRIDMenu)

/*  ------------------------------------------------------------------------  */

/*: AutoSimplification
        Define rules for Automatic Simplification  */

AutoSimplification::\
   If[~P[_AutoSimp[Loaded]=1],\
     (Pr["Loading Automatic Simplifier ..."];\
      <"SEI$USR:[LSS.SMP.TOOLS]AUTO_SIMP.SMP")\
/* EndIf */ ]

/*  ------------------------------------------------------------------------  */
```

```
/*: NoAutoSimplification
        Turn off rules for Automatic Simplification  */

NoAutoSimplification::\
   If[~P[_NoAutoSimp[Loaded]=1],\
     (Pr["No more automatic simplification."];\
      <"SEI$USR:[LSS.SMP.TOOLS]NO-AUTO_SIMP.SMP")\
/* EndIf */ ]

/*  ------------------------------------------------------------------------  */

/*: GreenMenu
        Execute the Main Menu for computing the Green's Function  */

GreenMenu::Proc[\
   Lcl[%Menu1again,%Enteredvalue],\
\
   Initialize,\
\
   _ExitMenu[True]:0,\
\
   Lbl[%Menu1again],\
   MainMenu,   \
   If[%Enteredvalue:ENTERROUTINE,\
     /* true */\
       MainMenuNum[%Enteredvalue],\
     /* false */\
       Jmp[%Menu1again],\
     /*  expr */\
       Jmp[%Menu1again]  /*  endif */],\
\
   If[(%Enteredvalue = 99 | P[_ExitMenu[True]=1]), /* Then */\
      Ret[NORMALSMPROUTINE],\
/* Else */\
      Jmp[%Menu1again]\
/* EndIf */ ],\
\
/*  endproc */]


/* ------------------------------------------------------------------------  */


ROUTINEFORHELP::Proc[\
   Lcl[%Menu1again,%Enteredvalue],\
\
   ClearScreen,\
   Pr["

            * * * *  H E L P    M E N U  * * * *

Here are the sequence of steps to compute the Hybrid Green's Function:


  1)  Construct Model for Distributed Parameter System -> ComputeDPS

      1.1) Get dimension of distributed state model -> InputDPSdim
      1.2) Read in matrices ( Ad, Bd, SIGMA, GAMMA )
```

```
          for the model -> InputDPSmatrices
       1.3) Compute M(s,z) -> ComputeMsz
       1.4) Compute Hbc -> ComputeHbc
       1.5) Compute Green's Function for the Distributed
          Parameter System -> ComputeGreen

   "],\
CTOCONTINUE,\
Pr["

  2)  Construct Model for Lumped Parameter System -> ComputeLPS

      2.1) Get dimension of lumped state model -> InputLPSdim
      2.2) Accept matrices ( Al, Bl ) for this model -> InputLPSmatrices
      2.3) Compute resolvent R(s) -> ComputeRs
      2.4) Compute transfer function Hl(s) -> ComputeHls

  3)  Accept parmaters for incidence relation -> GetIncidenceRelation

  4)  Obtain and display frequency response model for hybrid state
      vector X -> ComputeHYBRID

      4.1) ... 4.8) Go through the sequence of steps to obtain the
                    the Hybrid Green's Function

  "],\
CTOCONTINUE,\
Pr["

  5)  Display result in terms of delayed exponentials -> CoPrimeResults

  6)  Generate FORTRAN code for any of the transfer
      functions computed  -> Generate FORTRAN code

  8)  Save results of current SMP session into DPSYS.RES, LPSYS.RES,
      and HYBRID.RES.  -> SaveDPSresults, SaveLPSresults, SaveHYBRIDresults


"],\
CTOCONTINUE,\
\
   Lbl[%Menulagain],\
   MainHELPMenu,   \
   If[%Enteredvalue:EnterRoutineForHELP,\
     /* true */\
       MainMenuHELPNum[%Enteredvalue],\
     /* false */\
       Jmp[%Menulagain],\
     /*  expr */\
       Jmp[%Menulagain]  /*  endif */],\
   If[%Enteredvalue ~= 99, Jmp[%Menulagain]  ], \
   Ret[]              /*  endproc */]


/* ------------------------------------------------------------------------ */
/*   UTILITIES                                                              */
/* ------------------------------------------------------------------------ */

ENTERROUTINE::Proc[\
/*       Simplifies to the value/expression entered                  */\
```

```
Lcl[%Enteredvalue],\
%Enteredvalue::Rd["Enter : ",],\
Ret[%Enteredvalue]  /*  endproc */]

/* ------------------------------------------------------------------------ */

EnterRoutineForHELP::Proc[\
/*       Simplifies to the value/expression entered                  */\
Lcl[%Enteredvalue],\
%Enteredvalue::Rd["Enter number for HELP : ",],\
Ret[%Enteredvalue]  /*  endproc */]


PRINTLINES[$n]::Do[%i,1,$n,,Pr[]]
/*         prints $n lines                                            */


CTOCONTINUE::Rd["Press <Enter> or <Return> key to continue: "]
/*              pause after output                                    */

/* ------------------------------------------------------------------------ */

/*        Exit from menu environment         */

NORMALSMPROUTINE::Proc[\
Open[{,,{80,23}},0], /* return to normal   */\
Pr["Entering Line-Oriented SMP. "],\
Pr["To return to menu, type -> GreenMenu "],\
Null\
/* endproc */]

If[~P[_Abstract[Loaded]=1], /* Then */\
   (<"SEI$USR:[LSS.SMP.HELP]ABSTRACT.SMP";\
    Abstract)]

If[~P[_MainMenuHELP[Loaded]=1], /* Then */\
   (Pr["Loading Help Menu . . ."];\
    <"SEI$USR:[LSS.SMP.HELP]MAIN_MENU.SMP")]

If[~P[_GreenIntro[Loaded]=1], /* Then */\
    <"SEI$USR:[LSS.SMP.HELP]INTRO.SMP"]

Null /* end of file */
```

```
            /** Menu Kernal for Distributed Parameter System **/

/*K:   application menu; DPS; distributed state model */

/*A: Ajaipal S. Virdy */
/*S: Systems Engineering Inc */
/*D: March 27, 1986 */

/* ---------------------------------------------------------------------- */

/* GLOBAL SYMBOLS: routine names */


   DPSMenuNum[1] ::  DPSIntroduction
   DPSMenuNum[2] ::  InputDPSdim
   DPSMenuNum[3] ::  InputDPSmatrices
   DPSMenuNum[4] ::  ComputeMsz
   DPSMenuNum[5] ::  ComputeHbc
   DPSMenuNum[6] ::  ComputeGreen
   DPSMenuNum[7] ::  DisplayDPSVariables
   DPSMenuNum[8] ::  FORTRANDPSmodel
   DPSMenuNum[9] ::  SaveDPSresults
   DPSMenuNum[77] :: Null
   DPSMenuNum[88] :: ROUTINEFORHELP

/*
        The Main Distributed Parameter System Menu
*/

DisplayDPSMenu::(\
ClearScreen;\
Pr[\
"

  * * *  D I S T R I B U T E D   P A R A M E T E R   S Y S T E M  * * *

                 * * * *  M E N U  * * * *

  ------------------------------------------------------

    1  INTRODUCTION                      2  InputDPSdim

    3  InputDPSmatrices                  4  ComputeMsz

    5  ComputeHbc                        6  ComputeGreen

    7  DisplayDPSVariables               8  GenerateFORTRANCode

    9  SaveDPSresults                    77 Return to Main Menu

    88 Help                              99 Exit to SMP


"])


/* ---------------------------------------------------------------------- */
/*     DPSMenu loops till 99 is entered                */
```

```
DPSMenu::Proc[\
   Lcl[%DPSMenuagain,%Enteredvalue],\
\
   _ExitMenu[True]:0,\
\
   Lbl[%DPSMenuagain],\
   DisplayDPSMenu,   \
\
   If[%Enteredvalue:ENTERROUTINE,\
     /* true */\
       DPSMenuNum[%Enteredvalue],\
     /* false */\
       Jmp[%DPSMenuagain],\
     /*  expr */\
       Jmp[%DPSMenuagain]  /*  endif */],\
\
   Sel[ /* Begin Sel */\
      P[%Enteredvalue = 77],\
        Ret[],\
      P[%Enteredvalue = 99],\
        Ret[_ExitMenu[True]:1],\
      Jmp[%DPSMenuagain]\
   /* End Sel */],\
\
/*  endproc */]


/* ------------------------------------------------------------------------ */


DPSIntroduction::(\
ClearScreen;\
Pr[\
"



     This program computes the GREEN's Function using the formula:

                                                    -1
                GreenR = Hbc SIGMA M(s,w)

                GreenL = -Hbc GAMMA M(s,L-w)

    where

                                                          -1
                   Hbc = M(s,z)[ SIGMA + GAMMA M(s,L) ]

                                -1
                          Ad   [ s I - Bd ]z
          Msz = M(s,z) = e

    and the input parameters are the Nd x Nd matrices:

                Ad, Bd, SIGMA, and GAMMA.


                                                                  "];\
CTOCONTINUE)
```

```
/* ------------------------------------------------------------------------ */

/*: InputDPSdim
      Get the dimension of the state space equation for the Distributed
      Parameter System and the dimension of the input vector (See notes)
                                                                          */

InputDPSdim::Proc[\
ClearScreen,\
Pr["
    The state space model for the Distributed Parameter System is

          .
          Xd(t,z) = Ad Xd'(t,z) + Bd Xd(t,z)

          SIGMA Xd(t,0) + GAMMA Xd(t,L) = Emat Fd(t)

 "],\
\
  Lcl[%getNd,%getMd],\
\
Lbl[%getNd],\
 Pr[" "],\
 Pr["How many state variables Xd in the Distributed Parameter System (DPS)?"],\
 Nd:Rd["Enter Nd:  ",],\
 If[~P[Natp[Nd]], /* Then */\
   (Pr["Nd must be a positive integer."]; Jmp[%getNd])],\
\
  Lbl[%getMd],\
     Pr[" "],\
     Pr["What is the dimension of the DPS input vector Fd(t)?"],\
     Md:Rd["Enter Md:  ",],\
     If[~P[Natp[Md]], /* Then */\
       (Pr["Md must be a positive integer."]; Jmp[%%getMd])],\
\
Null\
]; /* End Proc */


/* ------------------------------------------------------------------------ */

/* Load Define function from TOOLS.DIR  */

If[~P[ Define[Loaded]=1], /* Then */\
   <"SEI$USR:[LSS.SMP.TOOLS]DEFINE.SMP"]

/* ------------------------------------------------------------------------ */

/*: ReadMatrix
      Get elements of all the matrices in the state-space model for
      the Distributed Parameter System (DPS) */

InputDPSmatrices::Proc[\
ClearScreen,\
Pr["
    The State Space Model for the Distributed Paramater System is

          .
          Xd(t,z) = Ad Xd'(t,z) + Bd Xd(t,z)
```

```
          SIGMA Xd(t,0) * GAMMA Xd(t,L) = Emat Fd(t)

where Ad, Bd, SIGMA, and GAMMA are Nd x Nd matrices and Emat is an Nd x Md
matrix.
 "],\
\
   /*  Local Variables and Labels  */\
\
   Lcl[%Azero,%Tmp,%response],\
\
   Ad:Bd:SIGMA:GAMMA:,                    /* Initialize variables */\
   Str:{"Bd","SIGMA","GAMMA"},            /* Initialize list of names */\
\
Lbl[%Azero],\
   Pr[" "],\
   Pr["Enter elements of matrix Ad"],\
   Ad:MRd[Nd,Nd],\
   Pr[" "],\
   Pr[Prmat[Ad]],\
\
   If[ Det[Ad]=0, /* Then */\
     (Pr["Matrix Ad must be invertible"];Jmp[%Azero]) ],\
\
/* Check if there are any parameters in Ad matrix */\
/* O L D   M E T H O D  ===>   Parameters[Ad],\  */\
   Define[Ad],\
\
Do[ %i, 1, 3, 1,\
  ( /* Begin Do */\
    Lbl[ %loop ];\
       Pr[" "];\
       Pr[Fmt[,"Is matrix ",Str[%i]," equal to zero? "]];\
       %response:Rdh[ ];\
\
       Sel[ /* Begin Sel */\
\
          P[In[%response,{'y,'Y,'yes,'Yes}]],\
           Set[%Tmp[%i],0], /* Set matrix to 0 */\
\
          P[In[%response,{'n,'N,'no,'No}]],\
           ( /* begin */\
             Pr[" "];\
             Pr["Enter elements of matrix ", Str[%i]];\
             Pr[" "];\
             %Tmp[%i]:MRd[Nd,Nd];\
             Pr[" "];\
             Pr[Prmat[%Tmp[%i]]];\
\
           /* Check if there are any parameters in these matrices */\
              Define[%Tmp[%i]]\
\
           ) /* end */,\
\
          Jmp[%loop]              /* Incorrect response */\
\
          ] /* End Sel */\
\
  )\
  ] /* End Do */,\
\
```

```
Bd:%Tmp[1],\
SIGMA:%Tmp[2],\
GAMMA:%Tmp[3],\
\
/* Now read in matrix Emat ( See Dr. Bennett's Notes ) */\
\
   Pr[" "],\
   Pr["Enter the elements of matrix Emat "],\
   Pr[" "],\
   Emat:MRd[Nd,Md],\
   Pr[" "],\
   Pr[Prmat[Emat]],\
\
Null\
\
] /* End Proc */

/* ------------------------------------------------------------------------ */


ComputeMsz::Proc[\
ClearScreen,\
Pr["

     Procedure to compute:

                                -1
                           Ad  [sI - Bd] z
     Msz = M(s,z) =       e

                                 -1
     This routine computes  Ad  [ s I - Bd ]z

     and sends this as an argument to MatExp1

  "],\
\
   Lcl[%temp,%exponent],\
\
/* Load MatExp1 for computing the transition matrix from TOOLS.DIR */\
   If[~P[_MatExp1[Loaded]=1], /* Then */\
      (Pr["Loading [LSS.SMP.TOOLS]MATRIX_EXP.SMP . . ."];\
       <"SEI$USR:[LSS.SMP.TOOLS]MATRIX_EXP.SMP")\
/* EndIf */ ],\
\
   If[~Valp[Msz], /* i.e. Msz not defined  Then */\
      Jmp[%ComputeMsz],\
/* Else */\
      (%res:Rdh["Msz has already been computed. Compute again? ",];\
        If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
           Jmp[%ComputeMsz],\
     /* Else */\
           Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeMsz],\
\
   %temp:s Ar[{Nd,Nd}]-Bd,\
   %exponent:Minv[Ad].%temp,\
```

```
\
   Pr[" "],\
   Pr["Computing M(s,z) ..."],\
   Msz:MatExpl[%exponent,S,z],\
\
   Pr[" "],\
   Pr["Msz:"],\
   Pr[Msz],\
\
   CTOCONTINUE,\
\
Null\
] /*  endproc  */


/* ------------------------------------------------------------------------ */

ComputeHbc::Proc[\
ClearScreen,\
Pr["

    Procedure to compute:

                                                    -1
       Hbc = M(s,z) [ SIGMA + GAMMA M(s,L) ]

    and

       Hu = Hbc Emat

 "],\
\
   Lcl[%MsL],\
\
   If[~Valp[Hbc], /* i.e. Hbc not defined  Then */\
      Jmp[%ComputeHbc],\
/* Else */\
      (%res:Rdh["Hbc has already been computed. Compute again? ",];\
        If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
           Jmp[%ComputeHbc],\
     /* Else */\
           Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeHbc],\
\
   %MsL:S[Msz,z->L],\
   Pr[" "],\
\
   Pr["Now computing Hbc ..."],\
   Pr[" "],\
\
   Hbc:Msz.Minv[SIGMA+GAMMA.%MsL],\
\
/* Hbc:MultSCsimp[SCsimp[Hbc]], */\
\
   Pr["Hbc: "],\
   Pr[" "],\
   Pr[Hbc],\
\
```

```
    CTOCONTINUE,\
\
    Pr[" "],\
    Pr["Hu:"],\
    Hu:Hbc.Emat,\
    Pr[Prmat[Hu]],\
\
    CTOCONTINUE,\
\
Null    /* endproc   */]

/* ------------------------------------------------------------------------ */


ComputeGreen::Proc[\
ClearScreen,\
Pr["

    Procedure to compute:

                                        -1
       GreenR = Hbc SIGMA [ M(s,w) ]
and
       GreenL = -Hbc GAMMA M(s,L - w)


 "],\
\
    Lcl[%Msw,%MsLw],\
\
    If[~Valp[GreenL]&~Valp[GreenR], /* GreenL and GreenR not defined Then */\
       Jmp[%ComputeGreen],\
/* Else */\
       (\
     %res:Rdh["Green's Function have already been computed. Compute again? ",];\
         If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
            Jmp[%ComputeGreen],\
      /* Else */\
            Ret[Null,2]\
      /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeGreen],\
\
    %Msw:S[Msz,z->w],\
    %MsLw:S[Msz,z->L-w],\
\
    Pr["Now computing GreenR and GreenL ..."],\
\
    GreenR:Hbc.SIGMA.Minv[%Msw],\
\
    GreenL:-Hbc.GAMMA.%MsLw,\
\
    Pr[" "],\
    Pr["GreenR: "],\
    Pr[GreenR],\
    Pr[" "],\
\
    CTOCONTINUE,\
\
```

```
    Pr["GreenL: "],\
    Pr[GreenL],\
    Pr[" "],\
\
    CTOCONTINUE,\
\
Null\
/*  endproc  */]


/* ------------------------------------------------------------------------ */

CoPrimeResult::Pr["Not implemented ... yet"]

/* ------------------------------------------------------------------------ */

DisplayDPSVariables::(\
  ClearScreen;\
  Pr["

                    DISTRIBUTED PARAMETER SYSTEM
                   ******************************

          FUNCTION      VARIABLES USED                  CALLS MADE TO
          **********    ****************                ****************

        InputDPSdim:    Nd, Md                              -

   InputDPSmatrices:    Ad, Bd, SIGMA, GAMMA, Emat          -

         ComputeMsz:    Msz                             MatExp1, LapI

         ComputeHbc:    Hbc, Hu                             -

       ComputeGreen:    GreenL, GreenR                      -


       "];\
  CTOCONTINUE)

/* ------------------------------------------------------------------------ */

FORTRANDPSmodel::(\
  ClearScreen;\
  Pr["

     This module generates FORTRAN code for a transfer function you specify.

     Note:  You must edit the resulting file ( i.e remove unnecessary zeros,
            rename variables, add comments, redefine function names, add CALL
            statements, etc )

     "];\
   GenerateFORTRANCode;\
  CTOCONTINUE)

SaveDPSresults::(\
    ClearScreen;\
    Pr["
```

```
        Saving all results for Distributed Parameter System . . .

      "];\
    Put[Nd,Md,Ad,Bd,SIGMA,GAMMA,Emat,Msz,Hbc,Hu,GreenL,GreenR,\
        DPSYS.RES];\
    Pr["

Results saved in DPSYS.RES

      "];\
CTOCONTINUE)


_DPSMenu[Loaded]:1
```

```
            /** Menu Kernal for Lumped Parameter System **/

/*K:   application menu; LPS; Lumped state model */

/*A: Ajaipal S. Virdy */
/*S: Systems Engineering Inc */
/*D: March 27, 1986 */

/* ---------------------------------------------------------------------- */

/* GLOBAL SYMBOLS: routine names */


   LPSMenuNum[1] ::  LPSIntroduction
   LPSMenuNum[2] ::  InputLPSdim
   LPSMenuNum[3] ::  InputLPSmatrices
   LPSMenuNum[4] ::  ComputeRs
   LPSMenuNum[5] ::  ComputeHIs
   LPSMenuNum[6] ::  Null
   LPSMenuNum[7] ::  DisplayLPSVariables
   LPSMenuNum[8] ::  FORTRANLPSmodel
   LPSMenuNum[9] ::  SaveLPSresults
   LPSMenuNum[77] :: Null
   LPSMenuNum[88] :: ROUTINEFORHELP

/*
        The Main Lumped Parameter Syster Menu
*/

DisplayLPSMenu::(\
ClearScreen;\
Pr[\
"

    * * * L U M P E D   P A R A M E T E R   S Y S T E M * * *

                * * * *  M E N U  * * * *

  ------------------------------------------------------------

    1  INTRODUCTION                    2  InputLPSdim

    3  InputLPSmatrices                4  ComputeRs

    5  ComputeHIs                      6  CoPrimeResults

    7  DisplayLPSVariables             8  GenerateFORTRANCode

    9  SaveLPSresults                  77 Return to Main Menu

    88 Help                            99 Exit to SMP


"])


/* ---------------------------------------------------------------------- */
/*     LPSMenu loops till 99 is entered                */
```

```
LPSMenu::Proc[\
   Lcl[%LPSMenuagain,%Enteredvalue],\
\
   Lbl[%LPSMenuagain],\
   DisplayLPSMenu,   \
\
   _ExitMenu[True]:0,\
\
   If[%Enteredvalue:ENTERROUTINE,\
     /* true */\
       LPSMenuNum[%Enteredvalue],\
     /* false */\
       Jmp[%LPSMenuagain],\
     /*  expr */\
       Jmp[%LPSMenuagain]  /*  endif */],\
\
   Sel[ /* Begin Sel */\
      P[%Enteredvalue = 77],\
        Ret[],\
      P[%Enteredvalue = 99],\
        Ret[_ExitMenu[True]:1],\
      Jmp[%LPSMenuagain]\
   /* End Sel */],\
\
/*  endproc */]

/* ------------------------------------------------------------------------ */

LPSIntroduction::(\
ClearScreen;\
Pr[\
"
    The State Space Model for the Lumped Parameter System is

          .
          Xl(t) = Al Xl(t) + Bl Fl

    The Resolvent and Transfer function for the

    Lumped Parameter System, given the input matrices Al and Bl, is

                                  -1
              Rs = [ s I - Al ]          ( the Resolvent )

              Hls = Rs Bl                ( the Transfer function )

                                                                    "];\
CTOCONTINUE)

/* ------------------------------------------------------------------------ */

INPUTMATRIXDIMENSION[$row,$col]::Proc[\
/*  Local Variables */\
   Lcl[%again],\
Lbl[%again],\
$row:Rd["Enter the number of rows in the matrix: ",],\
```

```
$col:Rd["Enter the number of columns in the matrix: ",],\
\
If[(~P[Natp[$col]] | ~P[Natp[$row]]),(Pr[\
"Both row and column numbers must be positive integers."];Pr[];\
Jmp[%again])],\
\
Ret[($row;$col)]  /*  endproc  */]


/* ------------------------------------------------------------------------ */

InputLPSdim::Proc[\
ClearScreen,\
Pr["

   The Lumped Parameter State Space Model is

          .
          Xl(t) = Al Xl(t) + Bl Fl . .

  "],\
\
/* Local Variables */\
\
   Lcl[%getAl,%Row,%Col],\
\
Lbl[%getAl],\
\
Pr[],\
Pr["For matrix Al: "],\
Pr[],\
INPUTMATRIXDIMENSION[%Row,%Col],\
If[%Row~=%Col,(Pr["Matrix A must be a square matrix."];Jmp[%getAl])],\
\
Nl:%Row,  /* Note: %Row = %Col */\
\
Pr["For matrix Bl:"],\
Pr[],\
INPUTMATRIXDIMENSION[%Row,Ml],\
\
/* Note:  no. of rows of Al must equal no. of rows of Bl */\
\
If[%Row~=Nl,(%Row:Nl;Pr["No. of rows of Bl must be same as Al's"])],\
\
CTOCONTINUE,\
\
Null  /*  endproc  */]

/* ------------------------------------------------------------------------ */

InputLPSmatrices::Proc[\
ClearScreen,\
Pr["

   The Lumped Parameter State Space Model is

          .
          Xl(t) = Al Xl(t) + Bl Fl

  "],\
\
Pr[],\
```

```
\
Pr["Enter elements of matrix A1"],\
Pr[],\
A1:MRd[N1,N1],\
Define[A1],\
Pr[Prmat[A1]],\
\
CTOCONTINUE,\
\
Pr[],\
Pr["Enter elements of matrix B1"],\
Pr[],\
B1:MRd[N1,M1],\
Pr[],\
Define[B1],\
Pr[Prmat[B1]],\
Pr[],\
\
CTOCONTINUE,\
\
Null /* endProc */]

/* ------------------------------------------------------------------------ */

ComputeRs::Proc[\
ClearScreen,\
Pr["
    Procedure to compute the Resolvent:

                                  -1
             Rs = [ s I - A1 ]


 "],\
\
   If[~Valp[Rs], /* i.e. Rs not defined  Then */\
      Jmp[%ComputeRs],\
/* Else */\
      (%res:Rdh["Rs has already been computed. Compute again? ",];\
        If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
            Jmp[%ComputeRs],\
     /* Else */\
            Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeRs],\
\
    Pr["Computing Rs ... "],\
    Pr[],\
    Rs:Minv[s Ar[Dim[A1]]-A1],\
    Pr[Prmat[Rs]],\
    Pr[],\
\
    CTOCONTINUE,\
\
    Null  /*  endproc  */]


/* ------------------------------------------------------------------------ */
```

```
ComputeHis::Proc[\
ClearScreen,\
Pr["
    Procedure to compute the Transfer function:


                Hls = Rs Bl


 "],\
\
\
   If[~Valp[His], /* i.e. His not defined  Then */\
      Jmp[%ComputeHis],\
/* Else */\
      (%res:Rdh["His has already been computed. Compute again? ",];\
       If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
          Jmp[%ComputeHis],\
     /* Else */\
          Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeHis],\
\
    Pr[],\
    Pr[" His "],\
    His:Rs.Bl,\
    Pr[Prmat[His]],\
    Pr[],\
\
    CTOCONTINUE,\
\
    Null   /*  endproc  */]


/* ------------------------------------------------------------------------ */

CoPrimeResults::Pr["Not implemented ... yet"]

/* ------------------------------------------------------------------------ */

DisplayLPSVariables::(\
  ClearScreen;\
  Pr["
```

LUMPED PARAMETER SYSTEM

FUNCTION	VARIABLES USED	CALLS MADE TO
InputLPSdim:	Nl, Ml	-
InputLPSmatrices:	Al, Bl	-

```
          ComputeRs:    Rs                                -

          ComputeHis:   His                               -


        "];\
  CTOCONTINUE)

/* ------------------------------------------------------------------------ */

FORTRANLPSmodel::(\
  ClearScreen;\
  Pr["

     This module generates FORTRAN code for a transfer function you specify.

     Note:  You must edit the resulting file ( i.e. remove unnecessary zeros,
            rename variables, add comments, redefine function names, add CALL
            statements, etc )


      "];\
    GenerateFORTRANCode;\
    CTOCONTINUE)

/* ------------------------------------------------------------------------ */

SaveLPSresults::(\
    ClearScreen;\
    Pr["

    Saving all results for Lumped Parameter System . . .

     "];\
    Put[NI,MI,AI,BI,Rs,His,LPSYS.RES];\
    Pr["

Results saved in LPSYS.RES

      "];\
CTOCONTINUE)

_LPSMenu[Loaded]:1
```

```
            /** Menu Kernal for HYBRID System **/

/*K:   application menu; HYBRID; distributed state model */

/*A: Ajaipal S. Virdy */
/*S: Systems Engineering Inc */
/*D: March 27, 1986 */

/* ---------------------------------------------------------------------- */

/* GLOBAL SYMBOLS: routine names */


   HYBRIDMenuNum[1] ::  HYBRIDIntroduction
   HYBRIDMenuNum[2] ::  ComputeQs
   HYBRIDMenuNum[3] ::  ComputeQtildas
   HYBRIDMenuNum[4] ::  ComputeP2sw
   HYBRIDMenuNum[5] ::  SeperateQ1Q2s.
   HYBRIDMenuNum[6] ::  ComputeHYBRIDRsz
   HYBRIDMenuNum[7] ::  ComputeHYBRIDHbcTilda
   HYBRIDMenuNum[8] ::  ComputeHYBRIDGreenFnc
   HYBRIDMenuNum[9] ::  DisplayHYBRIDVariables
   HYBRIDMenuNum[10] :: GenerateFORTRANCode
   HYBRIDMenuNum[11] :: SaveHYBRIDresults
   HYBRIDMenuNum[77] :: Null
   HYBRIDMenuNum[88] :: ROUTINEFORHELP

/*
       The Main HYBRID System Menu
*/

DisplayHYBRIDMenu::(\
ClearScreen;\
Pr[\
*

          * * *  H Y B R I D   S Y S T E M  * * *

             * * * *  M E N U  * * * *


  ------------------------------------------------------

    1  INTRODUCTION                    2  ComputeQs

    3  ComputeQtildas                  4  ComputeP2sw

    5  SeperateQ1Q2s                   6  ComputeHYBRIDRsz

    7  ComputeHYBRIDHbcTilda           8  ComputeHYBRIDGreenFnc

    9  DisplayHYBRIDVariables          10 GenerateFORTRANCode

    11 SaveHYBRIDresults               77 Return to Main Menu

    88 Help                            99 Exit to SMP


*])
```

```
/* ------------------------------------------------------------------------ */
/*     HYBRIDMenu loops till 99 is entered                                  */

HYBRIDMenu::Proc[\
   Lcl[%HYBRIDMenuagain,%Enteredvalue],\
\
   _ExitMenu[True]:0,\
\
   Lbl[%HYBRIDMenuagain],\
   DisplayHYBRIDMenu,   \
\
   If[%Enteredvalue:ENTERROUTINE,\
     /* true */\
       HYBRIDMenuNum[%Enteredvalue],\
     /* false */\
       Jmp[%HYBRIDMenuagain],\
     /*  expr */\
       Jmp[%HYBRIDMenuagain]  /*  endif */],\
\
   Sel[ /* Begin Sel */\
      P[%Enteredvalue = 77],\
        Ret[],\
      P[%Enteredvalue = 99],\
        Ret[_ExitMenu[True]:1],\
      Jmp[%HYBRIDMenuagain]\
   /* End Sel */],\
\
/*  endproc */]


/* ------------------------------------------------------------------------ */


HYBRIDIntroduction::(\
ClearScreen;\
Pr[\
"
    This is the main program which computes the Green's function for the

    Hybrid System.  The Distributed System Green's function and Transfer

    function plus the Lumped System Resolvent and Transfer function must

    be computed before calling this routine.  Using those results and the

    General Incidence Relation, then we can proceed to compute the Hybrid

    System Green's Function and Resolvent.


                                                                    "];\
CTOCONTINUE)

/* ------------------------------------------------------------------------ */


ComputeQs::Proc[\
ClearScreen,\
```

```
Pr["

    Procedure to Compute the M x M augmented matrix Q(s):


            Qs = [ C3 His | C1 Hu(s,0) + C2 Hu(s,L) ]


 "],\
/*  Local Variables  */\
\
    Lcl[%mat1,%mat2,%Hus0,%HusL],\
\
   If[~Valp[Qs], /* i.e. Qs not defined  Then */\
      Jmp[%ComputeQs],\
/* Else */\
      (%res:Rdh["Qs has already been computed. Compute again? ",];\
       If[P[In[%res,{'y,'Yes,'yes,!Y}]], /* Then */\
          Jmp[%ComputeQs],\
     /* Else */\
          Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeQs],\
\
    Pr["Computing Qs..."],\
\
    %mat1:C3.His,\
\
    %Hus0:S[Hu,z->0],\
    %HusL:S[Hu,z->L],\
\
    %mat2:C1.%Hus0+C2.%HusL,\
\
/*  Augment the two matrices, %mat1 & %mat2, using the external
    function:  AUGMENTMATRIX                                        */\
\
    AUGMENTMATRIX[%mat1,%mat2,Qs],\
\
    Pr[],\
    Pr["Q(s): "],\
    Pr[Qs],\
    Pr[],\
\
    CTOCONTINUE,\
\
    Null  /* endproc */]

/* ------------------------------------------------------------------------ */

ComputeQtildas::Proc[\
ClearScreen,\
Pr["

    This Procedure computes the the following:

                                  |  ~    |
        ~                  -1     |  Q1(s)  |  MI
```

```
        Q(s) = [ Im + Q(s) ]     = | -------- |
                                   |  ~       |
                                   |  Q2(s)   |  Md
                                   |          |

                ~
        Qtildas = Q(s)

 "],\
\
/*  Local Variables  */\
\
    Lcl[%Im],\
\
   If[~Valp[Qtildas], /* i.e. Qtildas not defined  Then */\
      Jmp[%ComputeQtildas],\
/* Else */\
      (%res:Rdh["Qtildas has already been computed. Compute again? ",];\
       If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
          Jmp[%ComputeQtildas],\
     /* Else */\
          Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeQtildas],\
\
\
  Pr["Computing Qtildas..."],\
\
  %Im:Ar[Dim[Qs]],\
\
  Qtildas:Minv[%Im+Qs],\
\
  Pr[],\
  Pr["Qtildas: "],\
  Pr[],\
  Pr[Qtildas],\
  Pr[],\
\
  CTOCONTINUE,\
\
  Null\
\
/*  endproc  */]


/* ------------------------------------------------------------------------ */

ComputeP2sw::Proc[\
ClearScreen,\
Pr["

   Procedure to compute

      P2(s,w) = C1 GreenL(s,0,w) + C2 GreenR(s,L,w)

   where the results from the Distributed system are used here.
```

```
 "],\
\
/*  Local Variables  */\
\
  Lcl[%Grsow,%GrsLw,%temp],\
\
    If[~Valp[P2sw], /* i.e. P2sw not defined  Then */\
       Jmp[%ComputeP2sw],\
/* Else */\
       (%res:Rdh["P2sw has already been computed. Compute again? ",];\
         If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
            Jmp[%ComputeP2sw],\
      /* Else */\
            Ret[Null,2]\
      /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeP2sw],\
\
\
  Pr["Computing P2sw..."],\
\
  %Grsow:S[GreenL,z->0],\
\
  %GrsLw:S[GreenR,z->L],\
\
  P2sw:C1.%Grsow+C2.%GrsLw,\
\
  Pr[],\
  Pr["P2(s,w): "],\
  Pr[],\
  Pr[P2sw],\
  Pr[],\
\
  CTOCONTINUE,\
\
  Null  /*  endproc  */]

/* ------------------------------------------------------------------------ */

SeperateQ1Q2s::Proc[\
ClearScreen,\
Pr["

    Procedure to extract Q1s and Q2s from Qtildas

    Q1s is equal to the first Ml rows of Qtildas

    and Q2s is equal to the last Md rows of Qtildas


  "],\
\
/*  Local variables  */\
\
  Lcl[%colnum],\
\
    If[~Valp[Q1s]|~Valp[Q2s], /* i.e. Q1s and Q2s not defined  Then */\
       Jmp[%ComputeQ1Q2s],\
/* Else */\
```

```
      (%res:Rdh["Q1s and Q2s have already been computed. Compute again? ",];\
        If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
          Jmp[%ComputeQ1Q2s],\
      /* Else */\
          Ret[Null,2]\
      /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeQ1Q2s],\
\
\
  Pr["Computing Q1s and Q2s..."],\
\
  %colnum:Dim[Qtildas][2],\
\
  Do[i,1,Ml,1,Do[j,1,%colnum,1,Q1s[i,j]:Qtildas[i,j]]],\
\
  Do[i,1,Md,1,Do[j,1,%colnum,1,Q2s[i,j]:Qtildas[Ml+i,j]]],\
\
  Pr[],\
  Pr["Q1(s): "],\
  Pr[Q1s],\
  Pr[],\
\
  CTOCONTINUE,\
\
  Pr["Q2(s): "],\
  Pr[Q2s],\
  Pr[],\
\
  CTOCONTINUE,\
\
  Null  /*  endproc  */]


/* ------------------------------------------------------------------------ */


ComputeHYBRIDRsz::Proc[\
ClearScreen,\
Pr["

   Procedure to compute:


                |                          |
                |   [ I - Hls Q1s C3 ] Rs  |
        Rsz =   |--------------------------|
                |      -Hu Q2s C3 Rs       |
                |                          |


    Rsz is N x Nl, where N = Nd + Nl

  "],\
\
/* Local Variables */\
\
  Lcl[%tmp1,%tmp2,%tmp3],\
\
```

```
   If[~Valp[Rsz], /* i.e. Rsz not defined  Then */\
      Jmp[%ComputeRsz],\
/* Else */\
      (%res:Rdh["Rsz has already been computed. Compute again? ",];\
        If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
           Jmp[%ComputeRsz],\
     /* Else */\
           Ret[Null,2]\
      /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeRsz],\
\
\
  Pr["Computing Rsz..."],\
  Set[Rsz],\
\
  %tmp1:Ar[{N1,N1}]-H1s.Q1s.C3,\
  %tmp2:%tmp1.Rs,\
\
  %tmp3:-1 Hu.Q2s.C3.Rs,\
\
/* Now "join" the matrices %tmp2 and %tmp3 together */\
\
  JOINMATRIX[%tmp2,%tmp3,Rsz],\
\
  Pr[],\
  Pr["Rsz: "],\
  Pr[],\
  Pr[Rsz],\
  Pr[],\
\
  CTOCONTINUE,\
\
  Null  /*  endproc  */]

/* ------------------------------------------------------------------------ */

ComputeHYBRIDHbcTilde::Proc[\
ClearScreen,\
Pr["

    Procedure to compute hybrid HbcTilde.

                 | H1s   0  |
    HbcTilde  =  |          | Qtildas D
                 | 0     Hu |


 "],\
\
/*  Local Variables  */\
\
  Lcl[%zeroN1Nd,%zeroNdN1,%tmp1,%tmp2,%tmp3],\
\
   If[~Valp[HbcTilde], /* i.e. HbcTilde not defined  Then */\
      Jmp[%ComputeHbcTilde],\
/* Else */\
      (%res:Rdh["HbcTilde has already been computed. Compute again? ",];\
        If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
```

```
            Jmp[%ComputeHbcTilda],\
      /* Else */\
            Ret[Null,2]\
      /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeHbcTilda],\
\
\
  Pr["Computing HbcTilda..."],\
\
  %zeroNlMd:Ar[{Nl,Md},0],\
  %zeroNdMl:Ar[{Nd,Ml},0],\
\
  AUGMENTMATRIX[Hls,%zeroNlMd,%tmp1],\
  AUGMENTMATRIX[%zeroNdMl,Hu,%tmp2],\
\
  JOINMATRIX[%tmp1,%tmp2,%tmp3],\
\
  HbcTilda:%tmp3.Qtildas.D,\
\
  Pr[],\
  Pr["HbcTilda(s): "],\
  Pr[HbcTilda],\
  Pr[],\
\
  CTOCONTINUE,\
\
  Null  /*  endproc  */ ]

/* ------------------------------------------------------------------------ */

ComputeHYBRIDGreenFnc::Proc[\
ClearScreen,\
Pr["

   Procedure to compute:


                |                      |
                |   -Hls Q1s P2sw      |
     GRszwR =   |----------------------|
                | GreenR - Hu Q2s Rs   |
                |                      |


   and


                |                      |
                |   -Hls Q1s P2sw      |
     GRszwL =   |----------------------|
                | GreenL - Hu Q2s Rs   |
                |                      |



  "],\
\
/*  Local Variables  */\
```

```
\
  Lcl[%tmp1,%tmp2],\
\
    If[~Valp[GRszwL]|~Valp[GRszwR], /* Hybrid Green Fnc not defined  Then */\
       Jmp[%ComputeHYBRIDGreen],\
/* Else */\
       (\
%res:Rdh["GRszwL and GRszwR have already been computed. Compute again? ",];\
         If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
            Jmp[%ComputeHYBRIDGreen],\
      /* Else */\
            Ret[Null,2]\
      /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputeHYBRIDGreen],\
\
\
  Pr["Computing Hybrid Green's Function..."],\
\
  %tmp1:-1 His.Q1s.P2sw,\
\
  %tmp2:GreenR-Hu.Q2s.P2sw,\
\
/* Now "join" the matrices %tmp1 and %tmp2 together */\
\
  JOINMATRIX[%tmp1,%tmp2,GRszwR],\
\
  Pr[],\
  Pr["GRszwR: "],\
  Pr[],\
  Pr[GRszwR],\
  Pr[],\
\
  CTOCONTINUE,\
\
  %tmp2:GreenL-Hu.Q2s.P2sw,\
/* Now "join" the matrices %tmp1 and %tmp2 together again */\
\
  JOINMATRIX[%tmp1,%tmp2,GRszwL],\
\
  Pr[],\
  Pr["GRszwL: "],\
  Pr[],\
  Pr[GRszwL],\
  Pr[],\
\
  CTOCONTINUE,\
\
  Null  /*  endproc  */]


/* ------------------------------------------------------------------------ */

DisplayHYBRIDVariables::(\
  ClearScreen;\
  Pr["
                          HYBRID PARAMETER SYSTEM
                         ************************
```

```
            FUNCTION        VARIABLES USED                          CALLS MADE TO
            **********      ****************                        **************

             ComputeQs:     Qs, C1, C2, C3, His, Hu                 AUGMENTMATRIX

         ComputeQtildas:    Qtildas, Qs                               -

            ComputeP2sw:    P2sw, C1, C2, GreenL, GreenR              -

          SeperateQ1Q2s:    Q1s, Q2s, Ml, Md                          -

       ComputeHYBRIDRsz:    Rsz, Q1s, Q2s, C3, Rs, His, Hu          JOINMATRIX

ComputeHYBRIDHbcTilda:      HbcTilda, His, Hu                       AUGMENTMATRIX
                                                                      JOINMATRIX

ComputeHYBRIDGreenFnc:      GRszwL, GRszwR, Q1s, Q2s, P2sw,         JOINMATRIX
                            GreenL, GreenR, Hu, Rs

        "];\
  CTOCONTINUE)

/* ------------------------------------------------------------------------ */

FORTRANLPSmodel::(\
  ClearScreen;\
  Pr["

     This module generates FORTRAN code for a transfer function you specify.

     Note:  You must edit the resulting file ( i.e. remove unnecessary zeros,
            rename variables, add comments, redefine function names, add CALL
            statements, etc )


      "];\
    GenerateFORTRANCode;\
    CTOCONTINUE)

/* ------------------------------------------------------------------------ */


SaveHYBRIDresults::(\
    Pr["Saving all results for HYBRID System . . ."];\
    Put[M,P,C1,C2,C3,D,Qs,Qtildas,P2sw,Q1s,Q2s,\
        Rsz,GRszwL,GRszwR,HbcTilda,HYBRID.RES];\
    Pr["

Results saved in HYBRID.RES

      "];\
CTOCONTINUE)

<"SEI$USR:[LSS.SMP.TOOLS]AUGMENT_MATRIX.SMP"
<"SEI$USR:[LSS.SMP.TOOLS]JOIN_MATRIX.SMP"

_HYBRIDMenu[Loaded]:1
```

```
/* ------------------------------------------------------------------------ */

GetIncidRels::Proc[\
ClearScreen,\
Pr["
    Procedure to read in the Incidence Relations:

      F(s) + C1 Xd(s,0) + C2 Xd(s,L) + C3 XI(s) = D U(s)

    where

          C1 , C2 are (MI + Md) x Nd matrices

          C3 is an (MI + Md) x NI matrix

    and

          D is an (MI + Md) x P matrix.

   "],\
\
  M:MI+Md,\
\
  If[~Valp[Md]|~Valp[MI]|~Valp[Nd]|~Valp[NI], /* Then */\
      (Pr["
You must compute the Distributed Parameter System and
the Lumped Parameter System before this!

        "];\
       CTOCONTINUE;\
       Ret[,2])],\
\
  Pr[],\
  Pr["Enter the elements for matrix C1: "],\
  Pr[],\
  C1:MRd[M,Nd],\
  Pr[Prmat[C1]],\
\
  Pr[],\
  Pr["Enter the elements for matrix C2: "],\
  Pr[],\
  C2:MRd[M,Nd],\
  Pr[Prmat[C2]],\
\
  Pr[],\
  Pr["Enter the elements for matrix C3: "],\
  Pr[],\
  C3:MRd[M,NI],\
  Pr[Prmat[C3]],\
\
  Pr[],\
  P:Rd["Enter P, the number of inputs to the hybrid system: ",],\
  Pr[],\
  Pr["Enter the elements for matrix D: "],\
  Pr[],\
  D:MRd[M,P],\
  Pr[Prmat[D]],\
\
  Null  /*  endproc  */]
```

_IncidenceRel[Loaded]:1

```
            /** Menu Kernal for Defining Cost Definition **/

/*K:   application menu; Green; cost definition */

/*A: Ajaipal S. Virdy */
/*S: Systems Engineering Inc */
/*D: March 27, 1986 */

/* ---------------------------------------------------------------------- */

/* GLOBAL SYMBOLS: routine names */


   CostDefMenuNum[1] ::  CostDefIntroduction
   CostDefMenuNum[2] ::  DistAverage
   CostDefMenuNum[3] ::  WeightedAverage
   CostDefMenuNum[4] ::  PointControl
   CostDefMenuNum[5] ::  ComputeGs
   CostDefMenuNum[6] ::  ComputeTsw
   CostDefMenuNum[7] ::  ComputeUs
   CostDefMenuNum[8] ::  PowerSeriesExpansion
   CostDefMenuNum[9] ::  SaveCostDefresults
   CostDefMenuNum[77] :: Null
/*
        The Cost Definition Menu
*/

DisplayCostDefMenu::(\
ClearScreen;\
Pr[\
"
            * * *  C O S T   D E F I N I T I O N  * * *

                * * * *  M  E  N  U  * * * *


  ------------------------------------------------------------

    1  INTRODUCTION                      2  Distributed State Average

    3  Weighted State Average           4  Point Control

    5  ComputeGs                        6  ComputeTsw

    7  ComputeUs                        8  PowerSeriesExpansion

    9  SaveCostDefresults               77 Return to Main Menu

    88 Help                             99 Exit to SMP


"])


/* ---------------------------------------------------------------------- */
/*     CostDefMenu loops till 99 is entered                 */


CostDefMenu::Proc[\
   Lcl[%CostDefMenuagain,%Enteredvalue],\
\
```

```
    Lbl[%CostDefMenuagain],\
    DisplayCostDefMenu,   \
\
    _ExitMenu[True]:0,\
\
    If[%Enteredvalue:ENTERROUTINE,\
      /* true */\
        CostDefMenuNum[%Enteredvalue],\
      /* false */\
        Jmp[%CostDefMenuagain],\
      /*  expr */\
        Jmp[%CostDefMenuagain]  /*  endif */],\
\
    Sel[ /* Begin Sel */\
       P[%Enteredvalue = 77],\
         Ret[],\
       P[%Enteredvalue = 99],\
         Ret[_ExitMenu[True]:1],\
       Jmp[%CostDefMenuagain]\
    /* End Sel */],\
\
/*  endproc */]


/* ------------------------------------------------------------------------ */


CostDefIntroduction::(\
ClearScreen;\
Pr["
CostDefinition

      Procedure to accept operator C to define controlled output and
      to compute effective transfer function model CR(s;A)

   Examples of Cost Definition:

   1)  Distributed state average
                  /L                    /L
            1     |                     |
       C = ---    | * dz                | = Integral from 0 to L
            L     |                     |
                 /0                    /0
   2)  Weighted State average
              /L
              |
       C =    | N(z) * dz
              |
             /0
   3)  Point Control
              /L
              |
       C =    | Delta(z1 - z) * dz
              |
             /0
  "];\
\
CTOCONTINUE;\
\
Pr["
```

With the C operator defined we can compute the following transfer functions:

$$G(s) = \begin{vmatrix} I & 0 \\ 0 & C \end{vmatrix}^{\sim} H(s,z)$$

$$T(s,w) = \begin{vmatrix} I & 0 \\ 0 & C \end{vmatrix} (GRszwL + GRszwR)$$

$$U(s) = \begin{vmatrix} I & 0 \\ 0 & C \end{vmatrix}^{\sim} R(s,z)$$

"];\
\
CTOCONTINUE)
/* ---------- */

/* ---------- */

_CostDefMenu[Loaded]:1

```
            /** Menu Kernal for Distributed Parabolic System **/

/*K:   application menu; DPS; distributed state model; parabolic system */

/*A: Ajaipal S. Virdy */
/*S: Systems Engineering Inc */
/*D: May 13, 1986 */

/* ------------------------------------------------------------------------ */

/* GLOBAL SYMBOLS: routine names */


   PARAMenuNum[1] ::  PARAIntroduction
   PARAMenuNum[2] ::  InputPARAdim
   PARAMenuNum[3] ::  InputPARAmatrices
   PARAMenuNum[4] ::  ComputePHIsz
   PARAMenuNum[5] ::  ComputePMsz
   PARAMenuNum[6] ::  ComputePHbc
   PARAMenuNum[7] ::  ComputePGreen
   PARAMenuNum[8] ::  DisplayPARAVariables
   PARAMenuNum[9] ::  SavePARAresults
   PARAMenuNum[77] :: Null
   PARAMenuNum[88] :: ROUTINEFORHELP

/*
        The Main Distributed Parameter System Menu
*/

DisplayPARAMenu::(\
ClearScreen;\
Pr[\
"

  *** D I S T R I B U T E D   P A R A B O L I C   S Y S T E M ***

                **** M E N U ****

  ____________________________________________________

    1  INTRODUCTION                    2  InputPARAdim

    3  InputPARAmatrices               4  ComputePHIsz

    5  ComputePMsz                     6  ComputePHbc

    7  ComputePGreen                   8  DisplayPARAVariables

    9  SavePARAresults                 77 Return to Main Menu

    88 Help                            99 Exit to SMP


"])


/* ------------------------------------------------------------------------ */
/*     PARAMenu loops till 99 is entered                */
```

```
PARAMenu::Proc[\
   Lcl[%PARAMenuagain,%Enteredvalue],\
\
   _ExitMenu[True]:0,\
\
   Lbl[%PARAMenuagain],\
   DisplayPARAMenu,   \
\
   If[%Enteredvalue:ENTERROUTINE,\
     /* true */\
       PARAMenuNum[%Enteredvalue],\
     /* false */\
       Jmp[%PARAMenuagain],\
     /*  expr */\
       Jmp[%PARAMenuagain]  /*  endif */],\
\
   Sel[ /* Begin Sel */\
      P[%Enteredvalue = 77],\
        Ret[],\
      P[%Enteredvalue = 99],\
        Ret[_ExitMenu[True]:1],\
      Jmp[%PARAMenuagain]\
   /* End Sel */],\
\
/*  endproc */]


/* ------------------------------------------------------------------------ */


PARAIntroduction::(\
ClearScreen;\
Pr[\
"
This program is used to compute the Green's Function for Parabolic Systems.
Given X an n-vector and g an m-vector, the Parabolic PDE is defined by
                               2
         d                   d                     d
         - X(t,z)   =   A  ---   X(t,z)  +  B  -  X(t,z)  +  C X(t,z)
         dt                    2                   dz
                             dz
with boundary conditions
                               d                                        d
  SIGMA1 X(t,0)  +  SIGMA2  -  X(t,0)  +  GAMMA1 X(t,L)  +  GAMMA2  -  X(t,L)  =  D g(t).
                               dz                                       dz
                  ^
The solution in X(s,z) is given by

                  /L
         ^       (                                  ^
         X(s,z) = ) Gr(s,z,w) X(0,w) dw   +   PHbc(s,z) g(s).
                  )
                 /0
                                                                   "];\
CTOCONTINUE;\
Pr[\
"
The Green's function, Gr(s,z,w), is defined as
                                                    | 0  |
                      -M(s,z) GAMMA PHI(s,L-w) |       | , 0 <= z < w
```

```
                                                  | In |
        Gr(s,z,w) =                               |    |
                                                  | Ø  |
                    M(s,z) SIGMA PHI(s,-w)        |    | , w < z <= L
                                                  | In |
where
                    LAMBDA z
        PHI(s,z) = e

                  |  Øn                  In   |
        LAMBDA =  |                           |
                  |   -1                 -1   |
                  | -A   [ C - s In ]   -A   B |
                                                        -1
        M(s,z) = [ In, Ø ] PHI(s,z) [ SIGMA + GAMMA PHI(s,L) ]

        SIGMA = [ SIGMA1, SIGMA2 ];  GAMMA = [ GAMMA1, GAMMA2 ]

 "];\
CTOCONTINUE)

/* ------------------------------------------------------------------------ */

ParaStSp::(\
ClearScreen;\
Pr["
    The State Space Model for the Parabolic System is
                          2
         d                d                d
         - X(t,z)  =  A  ---  X(t,z) + B  - X(t,z) + C X(t,z)
         dt               2               dz
                         dz

                          d                                    d
  SIGMA1 X(t,Ø) + SIGMA2  - X(t,Ø) + GAMMA1 X(t,L) + GAMMA2    - X(t,L) = D g(t).
                          dz                                   dz

Note:
      X is an Np-vector, so A is Np x Np. SIGMA1, SIGMA2, GAMMA1, GAMMA2 are
      2Np x Np matrices.
      g is an Mp-vector, so D is 2Np x Mp.

 "])

/* ------------------------------------------------------------------------ */

/*: InputPARAdim
       Get the dimension of the state space equation for the Distributed
       Parameter System and the dimension of the input vector (See notes)
                                                                        */

InputPARAdim::Proc[\
ParaStSp,\
  Lcl[%getNp,%getMp],\
\
Lbl[%getNp],\
 Pr[" "],\
 Pr["How many state variables Xp in the Parabolic System (PS)?"],\
 Np:Rd["Enter Np:  ",],\
 If[~P[Natp[Np]], /* Then */\
```

```
    (Pr["Np must be a positive integer."]; Jmp[%getNp])],\
\
  Lbl[%getMp],\
     Pr[" "],\
     Pr["What is the dimension of the PS input vector g(t)?"],\
     Mp:Rd["Enter Mp:  ",],\
     If[~P[Natp[Mp]], /* Then */\
       (Pr["Mp must be a positive integer."]; Jmp[%%getMp])],\
\
Null\
]; /* End Proc */


/* ------------------------------------------------------------------------ */

/* Load Define function from TOOLS.DIR  */

If[~P[ Define[Loaded]=1], /* Then */\
   <"SEI$USR:[LSS.SMP.TOOLS]DEFINE.SMP"]

/* ------------------------------------------------------------------------ */

/*: ReadMatrix
        Get elements of all the matrices in the state-space model for
        the Distributed Parameter System (PARA) */

InputPARAmatrices::Proc[\
ParaStSp,\
\
   /*  Local Variables and Labels  */\
\
   Lcl[%Azero,%Tmp,%response],\
\
   /* Initialize variables */\
\
   Ap:Bp:Cp:SIGMA:GAMMA:,\
   SIGMA1:SIGMA2:GAMMA1:GAMMA2:,\
\
   /* Initialize list of names */\
\
   Str:{"Bp", "Cp","SIGMA1", "SIGMA2", "GAMMA1", "GAMMA2"},\
\
Lbl[%Azero],\
   Pr[" "],\
   Pr["Enter elements of matrix Ap"],\
   Ap:MRd[Np,Np],\
   Pr[" "],\
   Pr[Prmat[Ap]],\
\
   If[ Det[Ap]=0, /* Then */\
     (Pr["Matrix Ap must be invertible"];Jmp[%Azero]) ],\
\
/* Check if there are any parameters in Ap matrix */\
/* O L D   M E T H O D  ===>   Parameters[Ap],\  */\
   Define[Ap],\
\
Do[ %i, 1, Len[Str], 1,\
  ( /* Begin Do */\
    Lbl[ %loop ];\
       Pr[" "];\
```

```
        Pr[Fmt[,"Is matrix ",Str[%i]," equal to zero? "]];\
        %response:Rdh[ ];\
\
        Sel[ /* Begin Sel */\
\
           P[In[%response,{'y,'Y,'yes,'Yes}]],\
            ( If[%i>=3, /* Then */  %Tmp[%i]:0 Ar[{2 Np,Np}],\
                        /* Else */  %Tmp[%i]:0 Ar[{Np,Np}] ]\
            ),\
\
           P[In[%response,{'n,'N,'no,'No}]],\
            ( /* begin */\
              Pr[" "];\
              Pr["Enter elements of matrix ", Str[%i]];\
              Pr[" "];\
\
              If[%i>=3, /* Then */  %Tmp[%i]:MRd[2 Np,Np],\
                        /* Else */  %Tmp[%i]:MRd[Np,Np] ];\
\
              Pr[" "];\
              Pr[Prmat[%Tmp[%i]]];\
\
            /* Check if there are any parameters in these matrices */\
               Define[%Tmp[%i]]\
\
            ) /* end */,\
\
           Jmp[%loop]              /* Incorrect response */\
\
           ] /* End Sel */\
\
  )\
  ] /* End Do */,\
\
Bp:%Tmp[1],\
Cp:%Tmp[2],\
SIGMA1:%Tmp[3],\
SIGMA2:%Tmp[4],\
GAMMA1:%Tmp[5],\
GAMMA2:%Tmp[6],\
\
/* AUGMENT SIGMA1 and SIGMA2 to get SIGMA */\
AUGMENTMATRIX[SIGMA1,SIGMA2,SIGMA],\
\
/* AUGMENT GAMMA1 and GAMMA2 to get GAMMA */\
AUGMENTMATRIX[GAMMA1,GAMMA2,GAMMA],\
\
/* Now read in matrix D ( See Dr. Bennett's Notes ) */\
\
   Pr[" "],\
   Pr["Enter the elements of matrix D "],\
   Pr[" "],\
   D:MRd[2 Np,Mp],\
   Pr[" "],\
   Pr[Prmat[D]],\
\
Null\
\
] /* End Proc */
```

```
/* ------------------------------------------------------------------------ */

ComputePHIsz::Proc[\
ClearScreen,\
Pr["

     Procedure to compute:


                    LAMBDA z
        PHI(s,z) = e

     where

                 |     0                 In    |
        LAMBDA = |                             |
                 |  -1                   -1    |
                 | -A   [ C - sIn ]     -A   B |


  "],\
\
   Lcl[%temp,%exponent],\
\
/* Load MatExp2 for computing the transition matrix from TOOLS.DIR */\
   If[~P[_MatExp2[Loaded]=1], /* Then */\
      (Pr["Loading [LSS.SMP.TOOLS]MATRIX_EXP.SMP . . ."];\
       <"SEI$USR:[LSS.SMP.TOOLS]MATRIX_EXP.SMP")\
/* EndIf */ ],\
\
   If[~Valp[PHIsz], /* i.e. PHIsz not defined  Then */\
      Jmp[%ComputePHIsz],\
/* Else */\
      (%res:Rdh["PHIsz has already been computed. Compute again? ",];\
       If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
          Jmp[%ComputePHIsz],\
     /* Else */\
          Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputePHIsz],\
\
   %eye:Ar[Dim[Ap]],\
   AUGMENTMATRIX[0 %eye, %eye, %top],\
   %tmp1:-Minv[Ap],\
   %tmp2:Cp-s %eye,\
   %tmp3:%tmp1.%tmp2,\
   %tmp4:%tmp1.Bp,\
   AUGMENTMATRIX[%tmp3,%tmp4,%bot],\
\
   JOINMATRIX[%top,%bot,LAMBDA],\
\
   Pr[" "],\
   Pr["LAMBDA:"],\
   Pr[LAMBDA],\
\
   %exponent:LAMBDA z,\
\
```

```
   Pr[" "],\
   Pr["Computing PHI(s,z) ..."],\
   PHIsz:MatExp2[%exponent,lam,z],\
\
   Pr[" "],\
   Pr["PHIsz:"],\
   Pr[PHIsz],\
\
   CTOCONTINUE,\
\
Null\
] /* endproc */

/* ------------------------------------------------------------------------ */

ComputePMsz::Proc[\
ClearScreen,\
Pr["
```

Procedure to compute:

$$M(s,z) = [\ In,\ 0\ ]\ PHI(s,z)\ [\ SIGMA + GAMMA\ PHI(s,L)\ ]^{-1} \quad (\ Np \times 2Np)$$

where

$$PHI(s,z) = e^{LAMBDA\ z} \quad (\ 2Np \times 2Np\ )$$

end

$$LAMBDA = \begin{vmatrix} 0 & In \\ -A^{-1}\ [\ C - sIn\ ] & -A^{-1}\ B \end{vmatrix} \quad (\ 2Np \times 2Np\ )$$

```
  "],\
\
   Lcl[%temp,%exponent],\
\
   If[~Valp[PMsz], /* i.e. PMsz not defined  Then */\
      Jmp[%ComputePMsz],\
/* Else */\
      (%res:Rdh["PMsz has already been computed. Compute again? ",];\
        If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
           Jmp[%ComputePMsz],\
     /* Else */\
           Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputePMsz],\
\
   Pr[" "],\
   Pr["Computing PM(s,z) ..."],\
\
   %eye:Ar[{Np,Np}],\
   AUGMENTMATRIX[%eye,0 %eye, %left],\
   %PHIsL:S[PHIsz,z->L],\
```

```
    %tmp1:Minv[SIGMA+GAMMA.%PHIsL],\
    PMsz:%left.PHIsz.%tmp1,\
\
    Pr[" "],\
    Pr["PMsz:"],\
    Pr[PMsz],\
\
    CTOCONTINUE,\
\
Null\
] /*  endproc  */


/* ------------------------------------------------------------------------ */

ComputePHbc::Proc[\
ClearScreen,\
Pr["

    Procedure to compute:

       PHbc = PMsz D    ( Np x Mp matrix )


 "],\
\
   Lcl[%MsL],\
\
   If[~Valp[PHbc], /* i.e. PHbc not defined  Then */\
      Jmp[%ComputePHbc],\
/* Else */\
      (%res:Rdh["PHbc has already been computed. Compute again? ",];\
        If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
           Jmp[%ComputePHbc],\
     /* Else */\
           Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputePHbc],\
\
   Pr[" "],\
   Pr["Now computing PHbc ..."],\
   Pr[" "],\
\
   PHbc:PMsz.D,\
\
   Pr["PHbc: "],\
   Pr[" "],\
   Pr[PHbc],\
\
   CTOCONTINUE,\
\
   Pr[" "],\
   Pr["Hu:"],\
   Hu:PHbc.Emat,\
   Pr[Prmat[Hu]],\
\
   CTOCONTINUE,\
\
```

```
Null    /* endproc   */]

/* ------------------------------------------------------------------------ */


ComputePGreen::Proc[\
ClearScreen,\
Pr["

    Procedure to compute:

                                                  | Ø  |
        GreenL  =    -M(s,z) GAMMA PHI(s,L-w)     |    | , Ø <= z < w
                                                  | In |


                                                  | Ø  |
        GreenR  =     M(s,z) SIGMA PHI(s,-w)      |    | , w < z <= L
                                                  | In |



 "],\
\
   Lcl[%Msw,%MsLw],\
\
   If[~Valp[PGreenL]&~Valp[PGreenR], /* PGreenL and PGreenR not defined Then */\
      Jmp[%ComputePGreen],\
/* Else */\
      (\
    %res:Rdh["PGreen's Function have already been computed. Compute again? ",];\
       If[P[In[%res,{'y,'Yes,'yes,'Y}]], /* Then */\
          Jmp[%ComputePGreen],\
     /* Else */\
          Ret[Null,2]\
     /* EndIf */ ])\
/* EndIf */ ],\
\
Lbl[%ComputePGreen],\
\
   %PHIsw:S[PHIsz,z->-w],\
   %PHIsLw:S[PHIsz,z->L-w],\
\
   Pr["Now computing PGreenR and PGreenL ..."],\
\
   %eye:Ar[{Np,Np}],\
   JOINMATRIX[Ø %eye, %eye, %right],\
\
   PGreenR:PMsz.SIGMA.%PHIsw.%right,\
\
   PGreenL:-PMsz.GAMMA.%PHIsLw.%right,\
\
   Pr[" "],\
   Pr["PGreenR: "],\
   Pr[PGreenR],\
   Pr[" "],\
\
   CTOCONTINUE,\
\
   Pr["PGreenL: "],\
```

```
   Pr[PGreenL],\
   Pr[" "],\
\
   CTOCONTINUE,\
\
Null\
/*  endproc  */]


/* ------------------------------------------------------------------------ */

CoPrimeResult::Pr["Not implemented ... yet"]

/* ------------------------------------------------------------------------ */

DisplayPARAVariables::(\
  ClearScreen;\
  Pr["

                      DISTRIBUTED PARAMETER SYSTEM
                     ******************************


          FUNCTION      VARIABLES USED                  CALLS MADE TO
         **********    *****************               ***************

        InputPARAdim:    Np, Mp                              -

   InputPARAmatrices:   Ap, Bp, Cp,
                        SIGMA1, SIGMA2,
                        GAMMA1, GAMMA2,
                        SIGMA, GAMMA, D                     -

        ComputePHIsz:   PHIsz, LAMBDA, Ap, Bp, Cp       MatExp2, LapI

         ComputePMsz:   PMsz, PHIsz, SIGMA, GAMMA           -

         ComputePHbc:   PHbc, PMsz, D                       -

       ComputePGreen:   PGreenL, PGreenR,
                        PMsz, PHIsz, SIGMA, GAMMA           -


        "];\
  CTOCONTINUE)

/* ------------------------------------------------------------------------ */

FORTRANPARAmodel::(\
  ClearScreen;\
  Pr["

     This module generates FORTRAN code for a transfer function you specify.

     Note:  You must edit the resulting file ( i.e remove unnecessary zeros,
            rename variables, add comments, redefine function names, add CALL
            statements, etc )

     "];\
   GenerateFORTRANCode;\
```

```
  CTOCONTINUE)

SavePARAresults::(\
    ClearScreen;\
    Pr["

        Saving all results for Distributed Parameter System . . .

      "];\
    Put[Np,Mp,Ap,Bp,Cp,SIGMA1,SIGMA2,GAMMA1,GAMMA2,SIGMA,GAMMA,D,\
        PHIsz,PMsz,PHbc,PGreenL,PGreenR,\
        PARASYS.RES];\
    Pr["

Results saved in PARAYS.RES

      "];\
CTOCONTINUE)                          . .

<"SYS$USER:[LSS3140.SMP.TOOLS]AUGMENT_MATRIX.SMP"
<"SYS$USER:[LSS3140.SMP.TOOLS]JOIN_MATRIX.SMP"

_PARAMenu[Loaded]:1
```

Matrix Tools

- Matrix Exponential
- Augment Matrices
- Join Matrices
- Block Diagonalize Matrix
- Extract Matrix

```
/* Procedure to obtain the matrix exponential of the argument list  */
/*  argument is assumed to be a square matrix                       */
/*A Wm. Bennett/SEI                */
/*D August 9, 1985                 */
/*S SEI                            */
/*K matrix, exponential, laplace */
MatExp[$x]::Proc[Lcl[%d,%i,%x],\
 %d:Dim[$x],%x:$x,\
 Do[%i,1,%d[1],1,\
  %x[%i,%i]:%x[%i,%i]-s1 ],\
 Ret[ LapI[Minv[-%x],s1] ]]
```

```
/* Procedure to obtain the matrix exponential of the argument list  */
/*  argument is assumed to be a square matrix                       */
/*A Wm. Bennett/SEI                */
/*D August 31, 1985                */
/*S SEI                            */
/*K matrix, exponential, laplace */
MatExp[$x]:: e Ar[Dim[$x]]-$x;\
 Ex[Minv[%%]];\
 LapI[%%]
```

```
/*: MatExp1[$mat,$s,$t]
      Computes the exponential of a matrix via Laplace Transforms */

MatExp1[$mat_=Sqmatp[$mat]]::Proc[\
/*
   This routine takes an exponential of a matrix.
   (this routine written by Dr. William Bennett)    */\
\
/*          At                -1
   Note:   e  = LapI[ (sI - A)  , s, t ]  */\
\
/* Load external file XLapI */\
   If[~P[_XLapI[Loaded]=1], /* Then */\
     (Pr["Loading external file XLapI ..."]; <XLapI)],\
\
   /* v is a dummy variable for LapI */\
   LapI[Minv[$s*Ar[Dim[$mat]]-$mat],$s,$t],\
\
   /*  endproc  */]

/* ------------------------------------------------------------------------ */

/*: Matp[$expr]
      yields 1 if $expr represents a matrix (rank 2 tensor). */

      Matp[$expr] :: Fullp[$expr,2]

/*: Sqmatp[$expr]
      yields 1 if $expr represents a square matrix. */

      Sqmatp[$expr] :: Matp[$expr] & P[Dim[$expr][1] = Dim[$expr][2]]

/* ------------------------------------------------------------------------ */

/*: MatExp2
      Computes the exponential of a matrix via the Cayley-Hamilton
      theorem */

MatExp2[$mat_=Sqmatp[$mat],$lam,$t]::Proc[\
/*
  Procedure to compute the matrix exponential of a matrix

  using Cayley-Hamilton's theorem  */\
\
/* Load external file XLapI */\
   If[~P[_XLapI[Loaded]=1], /* Then */\
     (Pr["Loading external file XLapI ..."]; <XLapI)],\
\
/* Load external routine XMat4 */\
  If[~P[_XMat4[Loaded]=1], /* Then */\
    (Pr["Loading external file XMat4 ..."]; <XMat4)],\
\
  Lcl[%poly,%a,%p,%n,%i,%j,%psi],\
\
  %poly:Ex[ Charpol[$mat,$lam] ],\
  %n:Len[$mat],\
\
/* Compute the coefficients of the characteristic polynomial of $mat */\
  Do[%i, 1, %n-1, 1,\
      %a[%n-%i]:Coef[$lam^%i,%poly]], /* End Do */\
```

```
  %a[%n]:S[%poly,$lam->0],\
\
/* Now compute the %n polynomials as follows */\
  %p[%n]:1,\
  Do[%j, %n, 2, -1,\
      %p[%j-1]:$lam %p[%j] + %a[%n+1-%j] %p[%n]], /* End Do */\
\
/* Now compute %n inverse Laplace Transforms as follows */\
  Do[%k, 1, %n, 1,\
      %psi[%k]:LapI[%p[%k]/%poly,$lam,$t]], /* End Do */\
\
/* Finally, compute the exponential of the matrix */\
  %eAt:%psi[1] Ar[{%n,%n},$i=$j] + Sum[%psi[%k+1] Mpow[$mat,%k],{%k,1,%n-1}],\
\
/* endproc */]

/* ---------------------------------------------------------------------- */

_MatExp1[Loaded]:1
_MatExp2[Loaded]:1
```

```
                /**  Matrix character tests  **/

/*K: square matrix; symmetric matrix; antisymmetric matrix; diagonal matrix;
     projection matrix; matrix classes; matrix types */

/*A: S.Wolfram */
/*S: California Institute of Technology */
/*D: July 1981 */

/*: Matp[$expr]
     yields 1 if $expr represents a matrix (rank 2 tensor). */

     Matp[$expr] :: Fullp[$expr,2]

/*: Sqmatp[$expr]
     yields 1 if $expr represents a square matrix. */

     Sqmatp[$expr] :: Matp[$expr] & P[Dim[$expr][1] = Dim[$expr][2]]

/*: Symp[$expr]
     yields 1 if $expr represents a symmetric matrix. */

     Symp[$expr] :: Matp[$expr] & P[$expr = Trans[$expr]]

/*: Asymp[$expr]
     yields 1 if $expr represents an antisymmetric matrix. */

     Asymp[$expr] :: Matp[$expr] & P[$expr = -Trans[$expr]]


/*: Diagp[$expr]
     yields 1 if $expr represents a diagonal matrix. */

     Diagp[$expr] :: Matp[$expr] & \
                     P[$expr-Diag[LDiag[$expr]]=Ar[Dim[$expr,2],0]]

/*: Projecp[$expr]
     yields 1 if $expr represents a projection matrix. */

     Projecp[$expr] :: Matp[$expr] & Ex[$expr.$expr=$expr]


#I[67]::  Close["[.tools]Matrix_prop.smp"]
```

```
AUGMENTMATRIX[$x,$y,$z]::Proc[\
/*
   This Procedure augments an m x n matrix $x with an m x r matrix $y

   to produce an m x (n + r) matrix $z

   ( i.e.  $z = [ $x | $y ] )
                                                                  */\
\
/*  Local Variables  */\
\
    Lcl[%m,%n,%r],\
\
%m:Dim[$x][1],\
%n:Dim[$x][2],\
%r:Dim[$y][2],\
\
/*  Note:  Both matrices ($x,$y) must have m rows  */\
\
If[(%m~=Dim[$y][1]),Ret[Pr["Cannot augment these matrices."]]],\
\
\
/*  Begin augmenting  */\
\
    $z:,\
    $z:$x,\
\
    Do[i,1,%m,1,\
      (Do[j,%n+1,%r+%n,1,$z[i,j]:$y[i,j-%n]]\
      )\
    ],  /* enddo */\
\
    Null  /*  endproc  */]
```

```
<"[.tools]AUGMENT.SMP"
<"[.tools]JOIN.SMP"
BLOCKDIAGONALIZE[$mat1,$mat2,$result]::Proc[\
/*
  This procedure block diagonalizes two input matrices ($mat1 and $mat2)

  and puts the result in $result.

  CALLS are made to AUGMENT.SMP and JOIN.SMP

                                                                    */\
\
/* Local Variables */\
\
   Lcl[%zero1,%zero2,%tophalf,%bottomhalf],\
\
   %zero1:Ar[{Dim[$mat1][1],Dim[$mat2][2]},0],\
   %zero2:Ar[{Dim[$mat2][1],Dim[$mat1][2]},0],\
\
   AUGMENTMATRIX[$mat1,%zero1,%tophalf],\
   AUGMENTMATRIX[%zero2,$mat2,%bottomhalf],\
\
   JOINMATRIX[%tophalf,%bottomhalf,$result],\
\
   Null\
\
   /*  endproc  */]
```

```
                    /* EXTRACT MATRIX FROM "JOINED MATRIX */

/*K:  extracts a matrix from a "joined matrix"                              */

/*A:  Ajaipal S. Virdy    */
/*S:  S.E.I.              */
/*D:  October 31,1985     */

/* ------------------------------------------------------------------------ */

EXTRACTMATRIX[$mat,$row1,$row2,$newmat]::Proc[\
/*
    Procedure to extract a matrix from a larger matrix

    $mat is the larger matrix from which to extract $newmat

    from $row1 to $row2.  That is,

    $newmat =  $mat[$row1,] to $mat[$row2,].
                                                          */\
\
/*  Local variables  */\
\
  Lcl[%colnum],\
\
  %colnum:Dim[$mat][2],\
  %rownum:$row2-$row1+1,\
\
  Do[i,1,%rownum,1,Do[j,1,%colnum,1,$newmat[i,j]:$mat[$row1-1+i,j]]],\
\
  Null  /*  endproc  */]

/* ------------------------------------------------------------------------ */
```

```
JOINMATRIX[$x,$y,$z]::Proc[\
/*
   This Procedure joins an m x r matrix $x with an n x r matrix $y

   to produce an (m + n) x r matrix $z

   ( i.e.        | $x |
          $z = | -- |
               | $y |     )                                          */\
\
/*  Local Variables  */\
\
    Lcl[%m,%n,%r],\
\
%m:Dim[$x][1],\
%n:Dim[$y][1],\
%r:Dim[$x][2],\
\
/*  Note:  Both matrices ($x,$y) must have r columns  */\
\
If[(%r~=Dim[$y][2]),Ret[Pr["Cannot join these matrices."]]],\
\
\
/*  Begin joining  */\
\
    $z:,\
    $z:$x,\
\
    Do[j,1,%r,1,Do[i,%m+1,%m+%n,1,$z[i,j]:$y[i-%m,j]]],\
\
    Null  /*  endproc  */]
```

Simplification Definitions

```
                        /* Identities Table */

/*A:  Ajaipal S. Virdy    */
/*S:  S.E.I.              */
/*D:  March 18, 1986      */

E^$x :: Exp[$x]

Exp[$x]^$n :: Exp[$x $n]

$x/Exp[$y] :: $x Exp[-$y]

$x/($$x Exp[$y]) :: $x/$$x Exp[-$y]

Exp[$x] Exp[$$y] :: Exp[$x + $$y]

/*  Do NOT define these rules yet  . .

(Exp[$x] + Exp[-$x]) / 2 :: Cosh[$x]

(Exp[$x] - Exp[-$x]) / 2 :: Sinh[$x]

(Exp[$x]/2 + Exp[-$x]/2) :: Cosh[$x]

(Exp[$x]/2 - Exp[-$x]/2) :: Sinh[$x]

($n Exp[$x] - $n Exp[-$x]) :: 2 $n Sinh[$x]

($n Exp[-$x] - $n Exp[$x]) :: -2 $n Sinh[$x]

($n Exp[$x] + $n Exp[-$x]) :: 2 $n Cosh[$x]

($n/$d Exp[$x] - $n/$d Exp[-$x]) :: 2 $n/$d Sinh[$x]

($n/$d Exp[$x] + $n/$d Exp[-$x]) :: 2 $n/$d Cosh[$x]

($n Cosh[$x]^2 - $n Sinh[$x]^2) :: $n

($$a Cosh[$x]^2/$$b - $$a Sinh[$x]^2/$$b) :: $$a/$$b

(Cosh[$x]^2 - Sinh[$x]^2) :: 1

Sinh[$x] Sinh[$y] :: 1/2 (Cosh[$x+$y]-Cosh[$x-$y])

Sinh[$x] Cosh[$y] :: 1/2 (Sinh[$x+$y]+Sinh[$x-$y])

Cosh[$x] Cosh[$y] :: 1/2 (Cosh[$x+$y]+Cosh[$x-$y])

Sinh[-$x] :: -Sinh[$x]

Cosh[-$x] :: Cosh[$x]

/* 1/2 (Cosh[$x+$y]-Cosh[$x-$y]) :: Sinh[$x] Sinh[$y] */

/* 1/2 (Sinh[$x+$y]+Sinh[$x-$y]) :: Sinh[$x] Cosh[$y] */

/* 1/2 (Cosh[$x+$y]+Cosh[$x-$y]) :: Cosh[$x] Cosh[$y] */

Cosh[I $$x] :: Cos[$$x]
```

```
Sinh[I $$x] :: I Sin[$$x]

*/

_AutoSimp[Loaded]:1
_NoAutoSimp[Loaded]:0
```

```
/* Substitute these strings for getting Delayed terms in function */
STr[1]: Sinh[$x] -> (1-Exp[-2 $x])/(2 Exp[-$x])
STr[2]: Cosh[$x] -> (1+Exp[-2 $x])/(2 Exp[-$x])
STr[3]: (1-Exp[-2 $x])^2 -> (1-2 Exp[-2 $x]+Exp[-4 $x])
STr[4]: (1+Exp[-2 $x])^2 -> (1+2 Exp[-2 $x]+Exp[-4 $x])
```

```
FindFunc[$x] :: (Pos[$n,$x] ; Do[%1,1,Len[%%],1,\
                (Fnc:Proj[$x,%%[%i]];\
                Sel[Fnc='Exp,Pr["Exponent"],\
                    Fnc='Cosh,Pr["Cosh"],\
                    Fnc='Sinh,Pr["Sinh"],\
                    Pr["What"]])] ; Null )
```

```
                        /* Identities Table */

/*A:  Ajaipal S. Virdy    */
/*S:  S.E.I.              */
/*D:  October 16, 1985    */

STr_:Ldist

STr[1,1,1]:  E^$x -> Exp[$x]

STr[1,1,2]:  Exp[$x] -> E^$x

STr[2,1,1]:  (Exp[$x] + Exp[-$x]) / 2 -> Cosh[$x]

STr[2,1,2]:  (Exp[$x] - Exp[-$x]) / 2 -> Sinh[$x]

STr[2,1,3]:  (Exp[$x]/2 + Exp[-$x]/2) -> Cosh[$x]

STr[2,1,4]:  (Exp[$x]/2 - Exp[-$x]/2) -> Sinh[$x]

STr[2,1,5]:  ($n Exp[$x] - $n Exp[-$x]) -> 2 $n Sinh[$x]

STr[2,1,6]:  ($n Exp[-$x] - $n Exp[$x]) -> -2 $n Sinh[$x]

STr[2,1,7]:  ($n Exp[$x] + $n Exp[-$x]) -> 2 $n Cosh[$x]

STr[2,1,8]:  ($n/$d Exp[$x] - $n/$d Exp[-$x]) -> 2 $n/$d Sinh[$x]

STr[2,1,9]:  ($n/$d Exp[$x] + $n/$d Exp[-$x]) -> 2 $n/$d Cosh[$x]

STr[3,1,1]:  ($n Cosh[$x]^2 - $n Sinh[$x]^2) -> $n

STr[3,1,2]:  (Cosh[$x]^2 - Sinh[$x]^2) -> 1

STr[4,1,1]:  Cosh[$x] / Sinh[$x] -> Coth[$x]

STr[4,1,2]:  Sinh[$x] / Cosh[$x] -> Tanh[$x]

STr[5,1,1]:  Sinh[$x] Sinh[$y] -> 1/2 (Cosh[$x+$y]-Cosh[$x-$y])

STr[5,1,2]:  Sinh[$x] Cosh[$y] -> 1/2 (Sinh[$x+$y]+Sinh[$x-$y])

STr[5,1,3]:  Cosh[$x] Cosh[$y] -> 1/2 (Cosh[$x+$y]+Cosh[$x-$y])

STr[6,1,1]:  Sinh[-$x] -> -Sinh[$x]

STr[6,1,2]:  Cosh[-$x] -> Cosh[$x]

STr[7,1,1]:  -Sinh[$x] -> Sinh[-$x]

STr[7,1,2]:  Cosh[$x] -> Cosh[-$x]

STr[8,1,1]:  1/2 (Cosh[$x+$y]-Cosh[$x-$y]) -> Sinh[$x] Sinh[$y]

STr[8,1,2]:  1/2 (Sinh[$x+$y]+Sinh[$x-$y]) -> Sinh[$x] Cosh[$y]

STr[8,1,3]:  1/2 (Cosh[$x+$y]+Cosh[$x-$y]) -> Cosh[$x] Cosh[$y]

STr[9,1,1]:  Cosh[I $$x] -> Cos[$$x]
```

```
STr[9,1,2]:  Sinh[I $$x] -> I Sin[$$x]

STr[10,1,1]: Cos[I $x] -> Cosh[$x]
STr[10,1,2]: Sin[I $x] -> I Sinh[$x]
```

```
                        /* Identities Table */

/*A:  Ajaipal S. Virdy    */
/*S:  S.E.I.              */
/*D:  March 24, 1988      */

/* Erase all automatic simplification rules */

E^$x :

Exp[$x]^$n :

(Exp[$x] + Exp[-$x]) / 2 :

(Exp[$x] - Exp[-$x]) / 2 :

(Exp[$x]/2 + Exp[-$x]/2) :

(Exp[$x]/2 - Exp[-$x]/2) :

($n Exp[$x] - $n Exp[-$x]) :

($n Exp[-$x] - $n Exp[$x]) :

($n Exp[$x] + $n Exp[-$x]) :

($n/$d Exp[$x] - $n/$d Exp[-$x]) :

($n/$d Exp[$x] + $n/$d Exp[-$x]) :

($n Cosh[$x]^2 - $n Sinh[$x]^2) :

($$a Cosh[$x]^2/$$b - $$a Sinh[$x]^2/$$b) :

(Cosh[$x]^2 - Sinh[$x]^2) :

Sinh[$x] Sinh[$y] :

Sinh[$x] Cosh[$y] :

Cosh[$x] Cosh[$y] :

Sinh[-$x] :

Cosh[-$x] :

/* 1/2 (Cosh[$x+$y]-Cosh[$x-$y]) :

/* 1/2 (Sinh[$x+$y]+Sinh[$x-$y]) :

/* 1/2 (Cosh[$x+$y]+Cosh[$x-$y]) :

Cosh[I $$x] :

Sinh[I $$x] :

_NoAutoSimp[Loaded]:1
_AutoSimp[Loaded]:0
```

```
                        /** Simplification Rules **/

/K:  simplification; Sinh; Cosh; Exp; hyperbolic identities */

/*A:  Ajaipal S. Virdy  */
/*S:  S.E.I.            */
/*D:  March, 25, 1986   */

/*
      This file provides simplification rules which SMP does not automatically
apply.  The simplification rules are embeded in function calls rather than
assigning the rules explicitly (See AUTO_SIMP.SMP).  Assigning the rules
tends to slow SMP down considerably since the assignments are made on the
Plus, Pow, and Mult projections.  */

/* Exceptions are for these two only */

Sinh[-$x] :: -Sinh[$x]

Cosh[-$x] :: Cosh[$x]

Exp[$x]^$n :: Exp[$x $n]

$x/Exp[$y] :: $x Exp[-$y]

Exp[$x] Exp[$y] :: Exp[$x+$y]

$x/($$x Exp[$y]) :: $x/$$x Exp[-$y]

/*: SCsmp[$expr]
        simplifies Cosh[$x]^2-Sinh[$x]^2 to 1 in $expr  */

        SCsimp[$expr] :: \
           If[ In[Pow[Cosh[$x],2],$expr] & \
               In[Pow[Sinh[$x],2],$expr], /* Then */\
             Si[$expr,\
                Cosh[$x]^2-Sinh[$x]^2 -> 1,\
                $$a*Cosh[$x]^2-$$a*Sinh[$x]^2 -> $$a,\
                Cosh[$x]^2/$b-Sinh[$x]^2/$b -> 1/$b,\
                ($$a*Cosh[$x]^2)/$b-($$a*Sinh[$x]^2)/$b -> $$a/$b\
               ], /* End Si */\
        /* Else */\
             $expr,$expr\
        /* End If */  ]


/*: Expsimp[$expr]
        simplifies Exponentials to Sinh and Cosh */

        Expsimp[$expr] :: \
          If[ In[Exp[$x],$expr] & In[Exp[-$x],$expr], /* Then */\
            ($expr:Ex[$expr,,'Div];\
             Si[$expr,\
                 (Exp[$x] + Exp[-$x]) / 2 -> Cosh[$x],\
                 ($n*Exp[$x] + $n*Exp[-$x]) -> 2 $n*Cosh[$x],\
                 (Exp[$x] - Exp[-$x]) / 2 -> Sinh[$x],\
                 ($n*Exp[$x] - $n*Exp[-$x]) -> 2 $n*Sinh[$x],\
                 (Exp[$x]/2 + Exp[-$x]/2) -> Cosh[$x],\
                 ($n*Exp[$x]/2 + $n*Exp[-$x]/2) -> $n*Cosh[$x],\
                 (Exp[$x]/2 - Exp[-$x]/2) -> Sinh[$x],\
```

```
                ($n*Exp[$x]/2 - $n*Exp[-$x]/2) -> $n*Sinh[$x],\
                ($$a*Exp[$x]/$b + $$a*Exp[-$x]/$b) -> 2 $$a/$b Cosh[$x],\
                ($$a*Exp[$x]/$b - $$a*Exp[-$x]/$b) -> 2 $$a/$b Sinh[$x]\
              ]), /* End Si */\
        /* Else */\
           $expr,$expr\
        /* End If */ ]


/*: MultSCsimp[$expr]
        simplifies product of Sinh and Cosh using hyperbolic identities */

        MultSCsimp[$expr] :: \
             If[ In[Map[Mult,{'Cosh,'Sinh}],$expr], /* Then */\
               Si[$expr,\
                  Sinh[$x] Sinh[$y] -> 1/2 (Cosh[$x+$y]-Cosh[$x-$y]),\
                  Sinh[$x] Cosh[$y] -> 1/2 (Sinh[$x+$y]+Sinh[$x-$y]),\
                  Cosh[$x] Cosh[$y] -> 1/2 (Cosh[$x+$y]+Cosh[$x-$y])\
                ], /* End Si */\
          /* Else */\
             $expr,$expr\
          /* End If */ ]

/* O L D   M E T H O D:

             If[ In[Mult[Sinh[$x],Sinh[$y]],$expr] | \
                 In[Mult[Cosh[$x],Cosh[$y]],$expr] | \
                 In[Mult[Cosh[$x],Sinh[$y]],$expr], /* Then */\
              . . .                                          */


/*: AutoSimp[$expr]
        Determines which rules to apply to expression */

        AutoSimp[$expr]::\
            (Expsimp[$expr];SCsimp[%%];MultSCsimp[%%])
```

Miscellaneous Routines

- Power Series Expansion
- Reduce subexpressions
- Get external files
- Define properties to symbols
- Generate FORTRAN CODE

```
PowerSeries[$f_=If[S[Num[$f],s->0]=0 & S[Den[$f],s->0]=0,1,\
                PrHold[12,$f,,"No pole-zero cancellation"],\
                PrHold[12,$f,,\
                "Cannot determine if there is pole-zero cancellation"]],\
                $a,$b]::\
           (Lcl[%ps];\
                %ps:Rat[Ax[Ps[$f,s,$a,$b]]];\
                Ex[%ps,,'Div];\
                Cb[%%,{s,s^$i}]\
           )

PrHold_: Tier          /* prevents printing of multiple messages */

Linecount_: Tier

RefFunc_: Tier

PrHold[$n,$x,$$y] :: (Linecount[$n,1];Len[#I]; \
                If[~(P[Linecount[$n,1]=Linecount[$n,2]]&P[RefFunc[$n,$x]=$x]), \
                Pr[Fmt[$$y]];Linecount[$n,2]:Len[#I]; \
                RefFunc[$n,$x]:$x;0,0,0])
```

```
/*: Define[$ex]
      set the parameters of $ex to be of a certain type, i.e. Realp, Imagp,
      Nonnegative, etc.. */

    Define[$ex]::\
      (Lcl[%param];\
       %param:Rel[Cont[$ex]];\
       If[Len[%param]=0, /* Then */ Ret[],\
  /* Else */\
        (Do[%i,1,Len[%param],1,\
            ( /* Begin Do */\
             Pr[Fmt[,"Is ",%param[%i]," Real ?"]];\
             %res:Rdh[];\
             If[In[%res,{'y,'Y,'yes,'Yes}], /* Then */\
                Map[Realp[$x]:1,List[%param[%i]]]];\
             Pr[Fmt[,"Is ",%param[%i]," > 0 ?"]];\
             %res:Rdh[];\
             If[In[%res,{'y,'Y,'yes,'Yes}], /* Then */\
                Map[$x>0:1,List[%param[%i]]]]\
            )] /* End Do */\
        )\
  /* End If */ ]\
    )
```

```
redpar[$x,$y] :: Cb[Ex[Rat[$x],Null,' Div],{$y,$y^$n}]

reduce[$x]::Cb[Ex[Rat[$x],,'Div],{z1,z1^$n}]
mxthru[$expr,$x]::Ex[Num[$expr]•$x]/Ex[Den[$expr]•$x]
computeHs[$list] :: 1 + Trans[S[Gs,$list]].Gs

computePhiLps[$list] :: Trans[S[Gs,$list]].CRlps
computePhiDps[$list] :: Trans[S[Gs,$list]].CRdps
```

```
Partable[$x] :: (Pos[$n,$x] ; Do[%i,1,Len[%%],Null,Pr[%%[%i],Proj[$x,\
                %%[%i]]]] ; Null)
```

```
/*: GetXfile[$xfile,$myfile]
      copies an external file $xfile into a local file $myfile */

   GetXfile[$xfile,$myfile]::\
      ( /* Begin */\
        Open[$myfile];\
        Dap[$xfile,,1];\
        Close[$myfile]\
      ) /* End */
```

```
/*: Symparts[$x]
        produces a list of all subparts of $x that are discrete symbols */

        Symparts[$x]::Map[Ap[$x,$1],Pos[$y_=Symbp[$y]&~Heldp[$y],$x]]

/*: union[$list]
        deletes duplicate entries in a list $list
        leaving them in the order of first occurence */

union[$list]::\
     (Lcl[%u,%e,%l];\
      If[Len[$list]=0,Ret[]];\
      %l:$list;\
      %u:{};\
      Loop[,%e:%l[1];%u:Cat[%u,{%e}];%l:Del[%e,%l,2],Len[%l]>0];%u)

/*: Parameters[$expr]
        determines if there are any parameters in the expression $expr
        if so, then ask the user about the properties of the parameter
        (i.e nonnegative, real, purely imaginary, complex, etc) */

Parameters[$expr]::\
 (Lcl[%list];\
  %list:union[Symparts[$expr]];\
  If[Len[%list]=0,Ret[Null]];\
  Do[%i,1,Len[%list],1,\
    ( /* Begin */\
      Pr[Fmt[,"What are the properties of ",%list[%i]," ? "]];\
      Pr[Fmt[3,"Real","Complex","Imaginary","Nonnegative"]]\
    ) /* End */\
    ])
```

```
UnLoadMenus::(\
   _DPSMenu[Loaded]:;\
   _LPSMenu[Loaded]:;\
   _HYBRIDMenu[Loaded]:;\
   _PARAMenu[Loaded]:;)
```

```
 GenerateFortranCode::Proc[\
\
/* Local Variables */\
\
   Lcl[%sub,%outfile],\
\
/* Get name for output file */\
\
   Pr[],\
   %outfile:Rd["Name of output file to put FORTRAN code into? ",],\
\
   Pr["Name of SUBROUTINE and parameters (enclosed in double quotes) "],\
   Pr["Example: ""GRNFNC( S, Z, W, N )"""],\
   Pr[],\
   %sub:Rdh[,],\
\
   fnc:Rd["Generate FORTRAN code for which function? ",],\
\
/*
   %res:Rdh["Would you like to add any comments? ",],\
   If[In[%res,{'y,'Y,'yes,'Yes,'YES}], /* Then */\
      %Comments:Rdh
*/\
\
   Open[Impl[Expl[%outfile]]],\
\
   Pr[Fmt[,"      SUBROUTINE ",Impl[Expl[%sub]]]],\
   Prog[fnc,,,2],\
\
   Close[Impl[Expl[%outfile]]],\
\
Null  /*  endproc  */ ]
```

B Fortan code for spectral factorization and optimal gain computation

Fortran Code for Numerical Algorithms

- spectral factorization by frequency sampling
- distributed gain computation by numerical quadrature
- stability/performance evaluation by frequency response testing

```
      Program Spectral_Factorization
      Implicit None
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     Mod:  17-Jul-1986 /Tsa
C     Mod:  11-Sep-1986 /Tsa (Get Rid Of Basile)
C     Mod:  12-Nov-1986 /Tsa (Correct specification of limits of
C        integration for QAWC in SpecIn2.
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     Main Routine To Compute Spectral Factorization Of A Positive-Definite
C     Hermitian Transfer Function Matrix Generated By Calls To An External
C     Fortran Procedure Called Hfunc
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      Include 'Spec_Stack.inc'
      Real Hrel(Nc,Nc,Np),Himg(Nc,Nc,Np),W(Np),
     &     Frel(Nc,Nc,Np),Fimg(Nc,Nc,Np)
      Real Rrel(Nc,Nc,Np),Rimg(Nc,Nc,Np),Eps(Np)
      Real Epsabs,Eparel,Eps3
      Real Wmin,Wmax,Wh
      Double Precision Epsdamp
      Common/Parama/Epsdamp
      Integer Nctl,Npts,Ier,Neval,Nitn,Itmp(Nc),Ksamp,Is,I
      Real X1(Np),Y1(Np),X2(Np),Y2(Np)
      Double Precision Dr1(Nc,Nc),Di1(Nc,Nc),Dr2(Nc),Di2(Nc)
      Real Alim
      Complex C1(Nc),C2(Nc)
C--
C--   Start of program
C--
      Print *,' Relative & Absolute Precision:  Eparel Epsabs'
      Read *,Eparel,Epsabs
      Print 17,Eparel,Epsabs
      eps3=0.0
 17   Format(3x,2e15.5)
      Print *,' Limit of integration for quadrature: Alim'
      Read *,Alim
      print 21,Alim
 21   Format(3x,e15.5)
      Print *,' Damping Term Eps (=2*Zeta)'
      Read *,Epsdamp
      Print 18,Epsdamp
 18   Format(3x,E15.5)
      Print *,' Enter Spectrum Range Of Transfer Function : Wmin, Wmax'
      Read *, Wmin,Wmax
      Print 19,Wmin,Wmax
 19   Format(3x,2e15.5)
      Print *, 'Uniform/Logarithmic Sampling (0/1)'
      Read *, Ksamp
      Print *,' Number Of Points In Frequency Domain : Np'
      Read 20,Npts
 20   Format(I20)
C--   Compute Frequency Sampling Interval
      If (Ksamp .eq. 1) Then
C--   Use Logarithmic Sampling
         W(1) = Wmin
         W(Npts) = Wmax
         Wh = 10.0**(Alog10(Wmax/Wmin)/(Npts-1))
         Do Is=2,Npts-1
            W(Is) = W(Is-1)*Wh
         End Do
      Else
```

```
C--   Use Uniform Sampling
         Wh = (Wmax-Wmin)/(Npts-1)
         W(1) = Wmin
         Do Is=2,Npts
            W(Is) = W(Is-1)+Wh
         end do
      Endif

C--
C--   Initialization
C--
C--   Transfer Function Dimensions
      Nctl=1
      Do I=1,Npts
         Call Hfunc(W(I),Hrel(1,1,I))
         Himg(1,1,I)=0
      end do

c--      Print *,' Number Of Iterations : '
c--      Read 20,Nitn
c--      Print 20,Nitn
      Nitn=0

      Open(Unit=20,File='Intermediate.results',Shared,
     $     Form='Unformatted',Status='New')
      Close(20)

      Call Specfc2(Hrel,Himg,Nctl,Nc,Frel,Fimg,W,Npts,Stkr,X1,Y1,
     $     X2,Y2,Itmp,Rrel,Rimg,Eps,Nitn,Epsabs,Epsrel,Eps3,Neval,
     $     Ier,C1,C2,Dr1,Di1,Dr2,Di2,Alim)

      Write(6,*) 'Total Function Evaluations = ',Neval

      End
```

```
      Subroutine Dg_Integrand(Sf,Omega,W,Gs,P1sz,P2p2i,P1sl,Integrand)
      Implicit None
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     Compute Integrand For Computation Of Distributed Gain
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C Integrand Is: [1/Fe][Ge][T(S,W)]
C     Where * Denotes Hermitian Conjugate,
C     Sf Is The Inverse Anti-Causal Spectral Factor At Omega
C     T Is Green(S,Z=0.5,W)
C     S Is The Laplace Variable,
C     And W Is The Spatial Coordinate (Range 0-1).
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
      Common/Params/Eps
      Double Precision Eps,W
      Double Complex Zeta,Sf,Zarg
      Double Complex S
      Double Complex P1sz(2,2),P1sl(2,2),P2p2i(2,2)
      Double Complex P2slw(2,2),P2sw(2,2)
      Double Complex Z,Gs(2),Grnsfn(2,2),Integrand(2)
      Real Omega

      Zeta = Dcmplx(Eps/2.0)
      S = Dcmplx(0.0,Omega)

C--   Compute The Green'S Function At (S,Z=.5,W)
      Z=0.5

      If(Dreal(Z).le.w)Then
         Zarg=Dcmplx(1.d+00-W)
         Call S_Phi2(S,Zarg,Zeta,P2slw)
         Call S_Green1(P2p2i,P2slw,Grnsfn)
      Else
         Zarg=Dcmplx(-W)
         Call S_Phi2(S,Zarg,Zeta,P2sw)
         Call S_Green2(P1sz,P2p2i,P1sl,P2sw,Grnsfn)
      Endif

C--   Compute The Integrand
      Integrand(1)=Sf * (Gs(1)*Grnsfn(1,1)+Gs(2)*Grnsfn(2,1))
      Integrand(2)=Sf * (Gs(1)*Grnsfn(1,2)+Gs(2)*Grnsfn(2,2))

      Return
      End
```

```
      Program Distributed_Gain
      Implicit None
C--   Compute The Distributed Gain For The Bernoulli-Euler Beam
C--   Using The Inverse-Anticausal Spectral Factor Generated By
C--   Sf.

      Common/Params/Eps

      Include 'G.inc'

      Logical Magic
      Double Precision Pi,Eps,W,Integral(2)
      Real Omega(1000),H,Frin,Fiin
      Double Complex F(1000),Integrand(2,1000),Result(2)
      Double Complex Gs(2,1000)
      Double Complex P1sz(2,2,1000),P2p2i(2,2,1000),P1si(2,2,1000)
      Double Complex P2sz(2,2),P2si(2,2)
      Double Complex Zeta,Z,S
      Integer Npts,Spts,I,J,K,Ii,Kk,Kkk
      Double Complex Denom

      Pi = 2.d+00*Dabs(Datan2(1.0d+00,0.0d+00))
      Zeta = Dcmplx(Eps)

C--   Algorithm Is Indicated In Comments By 'O':

C--   O Read In Eps Which Was Used In Spectral_Factorization

      Type *,'Eps From Spectral_Factorization?'
      Read(5,40)Eps
 40   Format(F)

C--   O Create A New Data File To Receive The Distributed Gains

      Open(Unit=21,File='Dg.results',Shared, Form='Unformatted',
     $     Status='New')
      Close(21)

C--   O Create A New Data File To Observe Convergence Of The Integral

      Open(Unit=22,File='Integral.results',Shared, Form='Unformatted',
     $     Status='New')
      Close(22)

C--   O Read In The Ordered Pairs (Omega, F)
C--   Where Omega Is The Frequency And F Is The I-A Spectral Factor

      Open(Unit=20,File='Sf.results',Shared, Form='Unformatted',
     $     Status='Old')

      Type*,'Number Of Points:'
      Read(5,*)Npts

      Type *,'Loading (Omega,Sf), Computing Gstar And Phi Matrices.'
      Do K=1,Npts
C--   Data Is Stored As Reals, H Is Not Necessary
         Read(20) Omega(K),H,Frin,Fiin
C--   Integrand Is Approximately Real And Symmetric With Respect To Omega=0
         F(K) = Dcmplx(Frin,Fiin)
C--   Compute Gs At Each Frequency Point
```

```
            Call S_G(Omega(K),G)
            Gs(1,K) = Conjg(G(1))
            Gs(2,K) = Conjg(G(2))

C--   Later Computations Require P1sz,P2sz,P2sl,P1sl,P2sw At (Z=.5,W)
C--   In Fact, Both Green'S Fn Definitions Require P2sz.minv[P2sl]
C--   Since Z Is Fixed, And Only One Term (P2sw) Depends Upon W, We
C--   Compute The Rest Now.

            S = Dcmplx(0.0,Omega(K))
            Z = 0.5
            Call S_Phi1(S,Z,Zeta,P1sz(1,1,K))
            Call S_Phi2(S,Z,Zeta,P2sz)
            Z = 1.0
            Call S_Phi1(S,Z,Zeta,P1sl(1,1,K))
            Call S_Phi2(S,Z,Zeta,P2sl)

C--   Compute P2sz.minv[P2sl]

            Denom = P2sl(1,1) * P2sl(2,2) - P2sl(1,2) * P2sl(2,1)
            P2p2i(1,1,K) =
     $           (P2sz(1,1) * P2sl(2,2) - P2sz(1,2) * P2sl(2,1)) / Denom
            P2p2i(1,2,K) =
     $           (P2sz(1,2) * P2sl(1,1) - P2sz(1,1) * P2sl(1,2)) / Denom
            P2p2i(2,1,K) =
     $           (P2sz(2,1) * P2sl(2,2) - P2sz(2,2) * P2sl(2,1)) / Denom
            P2p2i(2,2,K) =
     $           (P2sz(2,2) * P2sl(1,1) - P2sz(2,1) * P2sl(1,2)) / Denom

         End Do
         Close(20)

C--   Now Everything Independent Of W Is In Memory. Do The Computations...

C--   0 Determine The Spatial Range And Sampling Interval

         Type*,'Number Of Spatial Samples?'
         Read(5,*)Spts

C--   0 For Each Spatial Point:

         Do Ii = 0,Spts-1
            W = Dble(Ii)/Dble(Spts-1)
            Magic = .false.
            If(Abs(W).lt.0.001.and.abs(W).gt.0.999)Magic=.true.
C--   Progress Report:
            Type *,'**** W = ',W

C--   0 Compute The Integrand For All Values Of Omega At This W.

            Do J=1,Npts
               Call Dg_Integrand(F(J),Omega(J),W,Gs(1,J),
     $              P1sz(1,1,J),P2p2i(1,1,J),P1sl(1,1,J),
     $              Result)
               Integrand(1,J)=Result(1)
               Integrand(2,J)=Result(2)
            End Do

C--   0 Integrate By Generalized Trapezoidal Rule
```

```
         Do I=1,2
            Integral(I)=0.
            Do K=1,Npts-1
               Integral(I) = Integral(I) +
     $              Dreal((Integrand(I,K)+Integrand(I,K+1))) *
     $              Dble(Omega(K+1)-Omega(K))

C--    0 Dump The Results If This Is The Magic Value Of W (Spatial)
               If(Magic)Then
                  If(I.eq.1)Then
                     Open(22,File='Integral.results',Shared,
     $                    Access='Append',Status='Old',
     $                    Form='Unformatted')
                     Write(22)
     $                    Dble(Omega(K)),
     $                    Integrand(1,K),
     $                    Integral(1)
                     Close(22)
                  Endif
               Endif

            End Do
C--    End Looping On Values Of Omega
C--    The Trapezoidal Rule Requires Dividing By Two.
C--    The Integrand In This Case Needs A Division By Pi Also.
            Integral(I)=Integral(I)/(2.d+00*Pi)
         End Do
C--    Looping On Components Of Integrand Vector

C--    0 Save And Continue

         Open(21,File='Dg.results',Shared, Access='Append',Status='Old',
     $        Form='Unformatted')
         Write(21)W,Integral(1),Integral(2)
         Close(21)

      End Do
C--   Looping On Spatial Points

      End
```

```
      program Achieved_Spectral_Factor
      implicit none
c--   compute the achieved spectral factor for the Bernoulli-Euler beam
c--   using the distributed gains computed by DG.

      common/params/eps
      DOUBLE COMPLEX zeta
      DOUBLE COMPLEX S
      DOUBLE COMPLEX Z
      double precision eps,w(1000),dg(2,1000)
      double precision omega(1000),omin,omax,omegah
      double complex f(1000),integral
      double complex hbc(2,1000)
      integer wpts,opts,i,j,k,ksamp,ii

c--   o read in EPS which was used in Spectral_Factorization

      type *,'EPS from Spectral_Factorization?'
      read(5,*)eps

c--   o create a new data file to receive the spectral factor

      open(unit=21,file='sfa.results',shared,
     &     form='unformatted',status='new')
      close(21)

c--   o read in the ordered pairs (w, dg(1,), dg(2,))
c--   where w is the spatial position and dg(i,j) is
c--   the distributed gain i at point w(j).

      open(unit=20,file='dg.results',shared,readonly,
     &     form='unformatted',status='old')

      type*,'Number of points along beam?'
      read(5,*)wpts

      type *,'Loading (w,k1,k2)'
      do i=1,wpts
c--   data is stored as reals
         read(20)w(i),dg(1,i),dg(2,i)
      end do

      close(20)

c--   o determine the frequency range and sampling interval

      type*,'Frequency min and max?'
      read *, omin,omax
      print *, 'Uniform/logarithmic sampling (0/1)?'
      read *,ksamp
      print *,' Number of points in frequency domain?'
      read 37,opts
 37   format(i20)
C--   o compute frequency sampling interval
      IF (KSAMP .EQ. 1) THEN    ! use logarithmic sampling
         Omega(1) = OMIN
         Omega(OPTS) = OMAX
         OmegaH = 10.0**(LOG10(OMAX/OMIN)/(OPTS-1))
         do ii=2,opts-1
            Omega(ii) = Omega(ii-1)*OmegaH
```

```
      end do
      ELSE                        ! use uniform sampling
         OmegaH = (OMAX-OMIN)/(OPTS-1)
         Omega(1) = OMIN
         do ii=2,opts
            Omega(ii) = Omega(ii-1)+OmegaH
         end do
      ENDIF

      zeta = DCMPLX(EPS/2.0)

      do i=1,opts
         S = DCMPLX(0.d+00,OMEGA(i))
c--   Compute Hbc at point W(j) at frequency Omega(i)
         do j=1,wpts
            z=dcmplx(w(j))
            call s_hbc(s,z,hbc(1,j))
            hbc(1,j) = hbc(1,j) / s
            hbc(2,j) = hbc(2,j) / s
         end do !j; W loop

c--   o integrate by generalized trapezoidal rule

         integral=0.
         do j=1,wpts-1
            integral=integral+
     $           ((dg(1,j)*hbc(1,j)+dg(2,j)*hbc(2,j)) +
     $           (dg(1,j+1)*hbc(1,j+1)+dg(2,j+1)*hbc(2,j+1)))
     $           * (w(j+1)-w(j))
         end do !looping on values of w
c--   the trapezoidal rule requires dividing by two.
         integral=integral/2.d+00
c--   the integral formula is (1+Int(0,1,DG*HBC)), so add 1.
         integral=integral+1.

c--   o save and continue

         open(21,file='sfa.results',shared,
     $        access='append',status='old',
     $        form='unformatted')
         write(21)omega(i),dreal(integral),dimag(integral)
         close(21)

      end do !i; Omega loop

      end
```

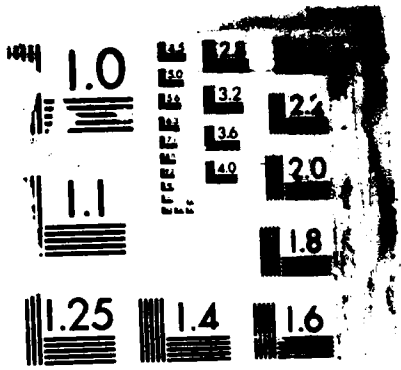
```
      Subroutine Specfc2(Hrel,Himg,Nctl,Ldh,
     $     Frel,Fimg,Omega,Npts,Stkr,
     $     X1,Y1,X2,Y2,Iwrk,Rrel,Rimg,Eps,Nitn,Eps1,Eps2,Eps3,
     $     Neval,Ier,C1,C2,Dxr,Dxi,Dx,Dy,Alim)
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     Compile With /D_Lines Options For Convergence Testing By
C     Storage Of Complex Functions In Matlab Compatible Format
C:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C--   External Variables
      Real Hrel(Ldh,Ldh,1),Himg(Ldh,Ldh,1),Omega(1),
     $     Frel(Ldh,Ldh,1),Fimg(Ldh,Ldh,1)
      Real Rrel(Ldh,Ldh,1),Rimg(Ldh,Ldh,1),Eps(1)
      Real Eps1,Eps2,Eps3,Stkr(1)
      Integer Nctl,Ldh,Npts,Ier,Neval,Nitn,Iwrk(1)
      Real X1(1),Y1(1),X2(1),Y2(1)
      Double Precision Dx(1),Dy(1),Dxr(Ldh,Ldh),Dxi(Ldh,Ldh)
      Complex C1(1),C2(1)
C--   Internal Workspace
      Real Work(8000)
      Integer Iwork(2000),Limit,Lenw
C--   System Timer/Statistics Support
      Integer*4 Timer_Addr, Status
      Integer*4 Lib$init_Timer, Lib$show_Timer
      External  Lib$init_Timer, Lib$show_Timer
      Integer Lfid(4),Lhid(4),Lrid(4),Idproj(4),Leid(4),Lfid1(4)
      Data Lfid, Idproj/'F','A','C','T','P','R','J','R'/
      Data Lhid,Lrid/'H',' ',' ',' ',' ','R','E','S','D'/
      Data Leid,Lfid1/'X','E','P','S','F','S','R','T'/
C-----------------------------------------------------------------
C     Author:  Wm.bennett/Sei                    Revised:  14-Mar-1986
C     Purpose: To Compute The Inverse Of The Anti-Causal Spectral Factor
C     Of Transfer Function (Assumed Positive Definite) From Frequency
C     Response Data.
C     On Entry:
C     Hrel              - Contains The Transfer Function Frequency Response
C     Nctl              - Dimension Of The Square Transfer Function
C     Npts              - Number Of Frequency Samples
C     Ldh               - Two Leading Dimensions Of The Arrays
C     Omega             - Contains The Real Frequency Data Samples
C     Stkr              - Main Stack Address
C     Nitn              - Maximum Number Of Iterations
C     Eps1              - Relative Precission For Numerical Integration
C     Eps2              - Absolute Precission For Numerical Integration
C     Eps3              - Residual Precission For Termination
C     On Return
C     Frel,Fimg         - Frequency Response Of Inverse Spectral Factor
C     Rrel,Rimg         - Residuals For Final Estimate
C     Ier               - Error Indicator For Numerical Integrator
C     Neval             - Number Of Integrand Evaluations
C     Eps               - Array Of Residual Norms For Each Iteration
C     Working Space
C     X1,Y1             - Arrays Of Length At Least Npts
C-----------------------------------------------------------------

      Status = Lib$init_Timer(Timer_Addr)
      If(.not. Status) Call Lib$signal(%val(Status))

C--   Compute Residuals As Shown Below
c--   Job = 0
```

```
C--   Compute Residuals In Reverse  3/15/86
      Job = 1

      Neval=0
      Ier=0
      Info = 0
      Limit=2000
      Lenw=Limit*4
      Inc = Ldh*Ldh
      Call Scal(Npts*Ldh*Ldh,0.0,Frel,1) !F:=0
      Call Scal(Npts*Ldh*Ldh,0.0,Fimg,1)

C--   Initialization

C--   Compute Spectral Factor Of Diagonal Elements
      Do J=1,Nctl
         Call Specin2(Hrel(J,J,1),Omega,Inc,Npts,Frel(J,J,1),
     $        Fimg(J,J,1),J,J,Alim,Eps1,Eps2,Limit,Lenw,
     $        Iwork,Work,Ie1,Nev1)
         Neval=Neval+Nev1
         Ier=Ier+Ie1
      end do

      Write(6,*) 'Function Evaluations For Initialization = ',Neval
      Status = Lib$show_Timer(Timer_Addr)
      If(.not. Status) Call Lib$signal(%val(Status))

C--   Algorithm Implementation

c--      Do Kk = 1,Nitn
c--C     Compute New Residuals And Davis' Termination Condition
c--         Call Resid(Hrel,Himg,Nctl,Ldh,Frel,Fimg,Npts,Rrel,Rimg,
c--     $        X1,Y1,C1,C2,Job)
c--
c--         Eps(Kk) = Epsres(Rrel,Rimg,Nctl,Ldh,Npts)
c--
c--         If(Eps(Kk).le.eps3) Goto 300
c--C     Projection Using Hilbert Transform
c--         Do Ii=1,Nctl
c--            Do Jj=1,Nctl
c--C--    .R:= Proj {R}
c--C--    .        +
c--               Call Hproj(Rrel(Ii,Jj,1),Rimg(Ii,Jj,1),Omega,
c--     $              Inc,Npts,X1,Y1,Stkr,X2,Y2,
c--     $              Alim,Eps1,Eps2,Limit,Lenw,
c--     $              Iwork,Work,Ie2,Nev2)
c--               Neval=Neval+Nev2
c--               Ier=Ier+Ie2
c--               Call Scopy(Npts,X1,1,Rrel(Ii,Jj,1),Inc)
c--               Call Scopy(Npts,Y1,1,Rimg(Ii,Jj,1),Inc)
c--            end do
c--         end do
c--
c--C     Compute New Inverse Spectral Factor By Davis' Iteration
c--
c--         Do Nw=1,Npts
c--            Do Ii=1,Nctl
c--               Rrel(Ii,Ii,Nw)=1.0+Rrel(Ii,Ii,Nw) !R:=I+R
c--            end do
c--            Call Newspc(Rrel(1,1,Nw),Rimg(1,1,Nw),Ldh,Dxr,Dxi,Nctl,
```

```
c--     &          Frel(1,1,Nw),Fimg(1,1,Nw),Ldh,Nctl,Dx,Dy,Iwrk,Info)
c--           If (Info .ne. 0) Goto 300
c--         end do
c--         Write(6,*) 'Current Function Evaluations = ',Neval
c--         Status = LibSshow_Timer(Timer_Addr)
c--         If(.not. Status) Call LibSsignal(%val(Status))
c--C--   Main Iteration Loop Of Davis' Algorithm
c--     end do
c--
c--C--   Compute New Residuals And Davis' Termination Condition
c--     Call Resid(Hrel,Himg,Nctl,Ldh,Frel,Fimg,Npts,Rrel,Rimg,
c--    &     X1,Y1,C1,C2,Job)
c--     Kk = Nitn
c--     Eps(Kk) = Epsres(Rrel,Rimg,Nctl,Ldh,Npts)
c--     Nitn = Kk
c--     Return

300  Continue
C--  Returns Spectral Factor In (Frel,Fimg)
C--  Residual In (Rrel,Rimg)
C--  Return Actual Number Of Iterations To Converge
     Nitn = Kk
     Return
     End
```

```
      REAL FUNCTION AH2(W,IC,JR,JOB)
      REAL W
      INTEGER IC1,JR1,JOB
      DATA IC1,JR1,JOB1/0,0,0/
C=====================================================================
C  Purpose: interface to standard QUADPACK routines
C  Variables:
C       W       - real frequency argument
C       IC      - column index
C       JR      - row index
C       JOB     - (0) => compute real part ELSE compute Imaginary part
C=====================================================================
      CALL HFUNC(W,HFUN)
      AH2 = alog(HFUN)
      RETURN
C   * * *  entry point for initialization
      ENTRY AHIN2(W,IC,JR,JOB)
       IC1 = IC
       JR1 = JR
       JOB1 = JOB
       AHIN2=0.0
      RETURN
      END
```

AD-A179 726 SPECTRAL FACTORIZATION AND HOMOGENIZATION METHODS FOR 3/3
MODELING AND CONTRO..(U) SYSTEMS ENGINEERING INC
GREENBELT MD W H BENNETT ET AL. 15 DEC 86 SEI-TR-86-14
UNCLASSIFIED AFOSR-TR-87-0502 F49620-84-C-0115 F/G 22/2 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
3.2
3.6
4.0

```
      SUBROUTINE CMTX(XR,XI,LDX,YR,YI,LDY,ZR,ZI,LDZ,M,N,L,CX,CY,JOB)
C       Wm.Bennett/SEI                        modifed: 14-MAR-1986
C ================================================================+
C  Purpose: to compute the product of two complex matrices and return
C the real and imaginary parts.
C       JOB=0   - compute usual matrix product
C                         Z(MxL) := X(MxN) * Y(NxL)
C       JOB>0   - compute matrix product as follows
C                                T
C                         Z(MxL) := X(MxN) * Y(NxL)
C       JOB<0   - compute matrix product as follows
C                                  T
C                         Z(MxL) := X(MxN) * Y(NxL)
C  External Functions called:
C       CDOTU
C       STOC
C ================================================================+
      REAL XR(LDX,LDX),XI(LDX,LDX),YR(LDY,LDY),YI(LDY,LDY)
      REAL ZR(LDZ,LDZ),ZI(LDZ,LDZ)
      INTEGER L,M,N,LDX,LDY,LDZ,JOB
      COMPLEX CX(N),CY(N),CTMP
      COMPLEX CDOTU,CDOTC

      IF (JOB) 300,1,200
1     DO 100 I=1,M
       DO 100 J=1,L
        CALL STOC(N,XR(I,1),XI(I,1),LDX,CX,1)
        CALL STOC(N,YR(1,J),YI(1,J),1,CY,1)
        CTMP = CDOTU(N,CX,1,CY,1)
        ZR(I,J) = REAL(CTMP)
        ZI(I,J) = AIMAG(CTMP)
100   CONTINUE
      RETURN

200   IF (N .NE. M) GOTO 500
      DO 250 I=1,M
       DO 250 J=1,L
        CALL STOC(N,XR(1,I),XI(1,I),1,CX,1)
        CALL STOC(N,YR(1,J),YI(1,J),1,CY,1)
        CTMP = CDOTC(N,CX,1,CY,1)
        ZR(I,J) = REAL(CTMP)
        ZI(I,J) = AIMAG(CTMP)
250   CONTINUE
      RETURN

300   IF (L .NE. N) GOTO 500
      DO 350 I=1,M
       DO 350 J=1,L
        CALL STOC(N,XR(I,1),XI(I,1),LDX,CX,1)
        CALL STOC(N,YR(J,1),YI(J,1),LDY,CY,1)
        CTMP = CDOTC(N,CY,1,CX,1)
        ZR(I,J) = REAL(CTMP)
        ZI(I,J) = AIMAG(CTMP)
350   CONTINUE
      RETURN

500   WRITE(6,*) 'INDEX ERROR IN CMTX'
      RETURN
      END
```

```
      REAL FUNCTION EPSRES(RR,RI,NC,LD,NPTS)
C            author: Wm. Bennett/SEI                revised: 3-Feb-1986
C=============================================================================+
C Purpose: To compute the residual measure:
C     max{ w: sum abs{R(i,j)} }
C=============================================================================+
      REAL RR(LD,LD,1), RI(LD,LD,1), X, XBIG
      INTEGER NC,LD,NPTS
      DATA XBIG/1.7E38/

      XMAX=-XBIG
      NC1=NC*NC
      DO 10 NW=1,NPTS
C * * * compute the sum of magnitudes of elements of (RR,RI) columnwise
       TRCR=0.
       TRCI=0.
       DO 5 JC=1,NC
        TRCR = SASUM(NC,RR(1,JC,NW),1) + TRCR
        TRCI = SASUM(NC,RI(1,JC,NW),1) + TRCI
5      CONTINUE
       X=SQRT(TRCR*TRCR+TRCI*TRCI)
       IF(X.GT.XMAX) XMAX=X
10    CONTINUE
      EPSRES = XMAX
      RETURN
      END
```

```
      Subroutine S_G(Omega,G)
      Implicit None
C*********************************************************************
C     Compute G(S)=Hbc(S,Z=L/2) For Computation Of Distributed Gain
C*********************************************************************
      Real Omega
      Double Complex S,Z
      Include 'Hbc.inc'
      Include 'G.inc'

      S = Dcmplx(0.0,Omega)

C     Compute (Hbc(S,Z=L/2))
      Z=.5
      Call S_Hbc(S,Z,Hbc)

      G(1)=Hbc(1)
      G(2)=Hbc(2)

      Return
      End
```

```
Subroutine S_Green1(P2p2i,P2s1w,Green1)
Implicit None

Double Complex P2p2i(2,2),P2s1w(2,2)
Double Complex Green1(2,2)

Green1(1,1) = - P2s1w(1,1) * P2p2i(1,1) - P2s1w(2,1) * P2p2i(1,2)
Green1(1,2) = - P2s1w(1,2) * P2p2i(1,1) - P2s1w(2,2) * P2p2i(1,2)
Green1(2,1) = - P2s1w(1,1) * P2p2i(2,1) - P2s1w(2,1) * P2p2i(2,2)
Green1(2,2) = - P2s1w(1,2) * P2p2i(2,1) - P2s1w(2,2) * P2p2i(2,2)

Return
End
```

```
      Subroutine S_Green2(P1sz,P2p2i,P1si,P2sw,Green2)
      Implicit None

      Double Complex P1sz(2,2),P2p2i(2,2),P1si(2,2),P2sw(2,2)
      Double Complex Green2(2,2)

      Double Complex U(2,2)

      U(1,1) =
     $     P1sz(1,1) - P2p2i(1,1) * P1si(1,1) - P2p2i(1,2) * P1si(2,1)
      U(1,2) =
     $     P1sz(1,2) - P2p2i(1,1) * P1si(1,2) - P2p2i(1,2) * P1si(2,2)
      U(2,1) =
     $     P1sz(2,1) - P2p2i(2,1) * P1si(1,1) - P2p2i(2,2) * P1si(2,1)
      U(2,2) =
     $     P1sz(2,2) - P2p2i(2,1) * P1si(1,2) - P2p2i(2,2) * P1si(2,2)

      Green2(1,1) = U(1,1) * P2sw(1,1) + U(1,2) * P2sw(2,1)
      Green2(1,2) = U(1,1) * P2sw(1,2) + U(1,2) * P2sw(2,2)
      Green2(2,1) = U(2,1) * P2sw(1,1) + U(2,2) * P2sw(2,1)
      Green2(2,2) = U(2,1) * P2sw(1,2) + U(2,2) * P2sw(2,2)

      Return
      End
```

```
      Subroutine Hfunc(Omega,H)
      Implicit None
C*****************************************************************************
C     Transfer Function H(W)=I+G'(-Iw)G(Iw) For Spectral Factorization.
C        H_Tilde = -S^2+G_Tilde'*G_Tilde = Omega^2 + G_Tilde' * G_Tilde
C*****************************************************************************
      Include 'G.inc'
      Real Omega,H

      Call S_G(Omega,G)

      H = Omega**2 + Conjg(G(1))*G(1)+Conjg(G(2))*G(2)

      Return
      End
```

```
      Subroutine S_Hbc(S,Z,Hbc)
      Implicit None

      Double Complex S,Z

      Include 'Hbc.inc'

      Common/Params/Eps
      Double Precision Eps
      Double Complex Zeta, L/1.0/
      Double Complex P2sz(2,2),P2sl(2,2),Denom

      Include 'Trig-Hyp.inc'

      Zeta = Dcmplx(Eps)

      Call S_Phi2(S,Z,Zeta,P2sz)
      Call S_Phi2(S,L,Zeta,P2sl)

      Denom = P2sl(1,1) * P2sl(2,2) - P2sl(1,2) * P2sl(2,1)
      If(Cdabs(Denom).lt.1.d-20)
     8     Type*,'%hbc-F-Zerden, Zero Denominator, Dummy!'

      Hbc(1) = (P2sz(1,1) * P2sl(1,2) - P2sz(1,2) * P2sl(1,1)) / Denom

      Hbc(2) = (P2sz(2,1) * P2sl(1,2) - P2sz(2,2) * P2sl(1,1)) / Denom

      Return
      End
```

```
      SUBROUTINE HPROJ(XREL,XIMG,W,INC,NPTS,FREL,FIMG,STKR,ZREL,ZIMG,
     &               ALIM,EPSREL,EPSABS,LIMIT,LENW,IWORK,WORK,IER,NEVAL)
C       DATE MODIFIED:  5-FEB-1988
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C  PURPOSE: to compute the Hilbert transform of a transfer function     *
C transfer function data contained in the arrays (XREL,XIMG)           *
C  External Variables:                                                 *
C       XREL, XIMG      - the real/imag parts of the transfer function *
C                         frequency response data                      *
C       W               - array of imaginary frequency data points     *
C       INC             - increment integer for nonsequential storage of *
C                         data (INC=0 if contiguous storage)           *
C       NPTS            - number of data points to be processed        *
C       EPSREL,EPSABS,  - relative/absolute precision for quadrature   *
C  On Return:                                                          *
C       FREL, FIMG      - the real/imag parts of the Hilbert transform *
C  External Refs:                                                      *
C       RFUNC1          - external function returns the real part of h(w) *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      EXTERNAL RFUNC1
C   * * * External Variables * * *
      REAL*4  XREL(1),XIMG(1),FREL(1),FIMG(1),W(1),ZREL(1),ZIMG(1)
      INTEGER IWORK(1)
      REAL*4  WORK(1),X,TMP
C   * * * Internal Variables * * *

      PI=4.*ATAN(1.)
C  * * *  Pass pointers to interpolation routine
      tmp = RFIN1(W,%LOC(STKR),%LOC(W),%LOC(XREL),4,INC,NPTS)
C  * * *  Compute the Hilbert transform for real part
      DO 100  K=1,NPTS
       CALL QAWC(RFUNC1,-ALIM,ALIM,W(K),EPSABS,EPSREL,X,ABSERR,NEVAL,
     &      IER,LIMIT,LENW,LAST,IWORK,WORK)
       ZREL(K) = X/PI
100   CONTINUE
C  * * *  Pass pointers to interpolation routine
      tmp = RFIN1(W,%LOC(STKR),%LOC(W),%LOC(XIMG),4,INC,NPTS)
C  * * *  Compute the Hilbert transform for imaginary part
      DO 200  K=1,NPTS
       CALL QAWC(RFUNC1,-ALIM,ALIM,W(K),EPSABS,EPSREL,X,ABSERR,NEVAL,
     &      IER,LIMIT,LENW,LAST,IWORK,WORK)
       ZIMG(K) = X/PI
200   CONTINUE
C  * * *  Compute the anti_causal projection (Frel,Fimg):= P_{(Xrel,Ximg)}
c      DO 300 K=1,NPTS
c       KK = (K-1)*INC+1
c       FREL(K) = (XREL(KK) + ZIMG(K))/2.
c300    FIMG(K) = (XIMG(KK) - ZREL(K))/2.
C  * * *  Compute the causal projection (Frel,Fimg):= P+{(Xrel,Ximg)}
      DO 300 K=1,NPTS
       KK = (K-1)*INC+1
       FREL(K) = (XREL(KK) - ZIMG(K))/2.
300    FIMG(K) = (XIMG(KK) + ZREL(K))/2.
      RETURN
      END
```

```
      SUBROUTINE NEWSPC(AR,AI,LDA,DAR,DAI,N,XR,XI,LDX,K,ZR,ZI,IPVT,INFO)
C           revised: 17-MAR-1986
C=====================================================================+
C Purpose: to solve the complex matrix equation
C                            -1
C                     X := X A
C On Entry:
C       AR,AI   - real/imag parts of NxN matrix A
C       LDA,LDX - leading dimension of arrays for AR,AI (resp. XR,XI)
C       XR,XI   - real/imag parts of KxN matrix X
C       ZR,ZI   - real working arrays of dimension at least max(N,K)
C       IPVT    - integer working array of dimension at least max(N,K)
C On return:
C       XR,XI   - contains the KxN solution
C       INFO    - if =0 then solution is correct to working precision
C=====================================================================+
      REAL AR(LDA,LDA),AI(LDA,LDA),XR(LDX,LDX),XI(LDX,LDX)
      DOUBLE PRECISION DAR(LDA,LDA),DAI(LDA,LDA),ZR(1),ZI(1),RCOND
      INTEGER IPVT(1), INFO

      INFO=0
      DO 1 J=1,N
       DO 1 I=1,N
        DAR(I,J) = AR(I,J)
1       DAI(I,J) = AI(I,J)
      CALL WGECO(DAR,DAI,LDA,N,IPVT,RCOND,ZR,ZI)
      IF (1.0+RCOND .EQ. 0.) GOTO 100
C  change to conjugate of LU factored DA
      DO 2 J=1,N
       DO 2 I=1,N
2       DAI(I,J) = - DAI(I,J)
C         solve K linear systems
      DO 30 I=1,K
       DO 10 J=1,N  !Get a row from (Xr,Xi)
        ZR(J) = XR(I,J)
10      ZI(J) = XI(I,J)
      CALL WGESL(DAR,DAI,LDA,N,IPVT,ZR,ZI,1)
      DO 20 J=1,N
       XR(I,J) = ZR(J)
       XI(I,J) = ZI(J)
20    CONTINUE
30    CONTINUE
      RETURN

100   WRITE(6,*) 'GAUSSIAN ELIMINATION FAILED IN NEWSPC'
      RETURN
      END
```

```
      Subroutine S_Phi1(S,Z,Zeta,Phi1)
      Double Complex S,Z,Zeta,Phi1(2,2)

      Double Complex Rs,Denom
      Double Complex C1,C1s
      Double Complex C2,C2s

      Include 'Trig-Hyp.inc'

      If(Abs(S).lt.1.0d-20) Then
         Phi1(1,1) = 1.
         Phi1(1,2) = 0.
         Phi1(2,1) = 0.
         Phi1(2,2) = 1.
         Return
      End If

      Rs = Cdsqrt(S)
      C1 = Cdsqrt(1. - Zeta*Zeta)
      C1s = -Zeta + I * C1
      C2s =  Zeta + I * C1
      C1 = Cdsqrt(C1s)
      C2 = Cdsqrt(C2s)
      Denom = C1s + C2s

      Phi1(1,1) = (C1s * Cos(C1 * Rs * Z) + C2s * Cosh(C2 * Rs * Z)) /
     $     Denom
      Phi1(1,2) = (Cos(C1 * Rs * Z) - Cosh(C2 * Rs * Z)) / Denom
      Phi1(2,1) = - Phi1(1,2)
      Phi1(2,2) = (C2s * Cos(C1 * Rs * Z) + C1s * Cosh(C2 * Rs * Z)) /
     $     Denom
      Return
      End
```

```
      Subroutine S_Phi2(S,Z,Zeta,Phi2)
      Double Complex S,Z,Zeta,Phi2(2,2)

      Double Complex Rs,Denom
      Double Complex C1,C1s
      Double Complex C2,C2s

      Include 'Trig-Hyp.inc'

      If(Abs(S).lt.1.0d-20) Then
         Phi2(1,1) = Z
         Phi2(1,2) = 0.
         Phi2(2,1) = 0.
         Phi2(2,2) = Z
         Return
      End If

      Rs = Cdsqrt(S)
      C1 = Cdsqrt(1. - Zeta*Zeta)
      C1s = -Zeta + I * C1
      C2s =  Zeta + I * C1
      C1 = Cdsqrt(C1s)
      C2 = Cdsqrt(C2s)
      Denom = Rs * (C1s + C2s)


      Phi2(1,1) = (C1 * Sin(C1 * Rs * Z) + C2 * Sinh(C2 * Rs * Z)) /
     $     Denom
      Phi2(1,2) = (Sin(C1 * Rs * Z) / C1 - Sinh(C2 * Rs * Z) / C2) /
     $     Denom
      Phi2(2,1) = - Phi2(1,2)
      Phi2(2,2) = (C2s * Sin(C1 * Rs * Z) / C1 +
     $     C1s * Sinh(C2 * Rs * Z) / C2) / Denom
      Return
      End
```

```
      SUBROUTINE RESID(HR,HI,N,LD,FR,FI,NPTS,RR,RI,XR,XI,C1WRK,C2WRK,
     &                 JOB)
      REAL HR(LD,LD,1),HI(LD,LD,1),FR(LD,LD,1),FI(LD,LD,1)
      REAL RR(LD,LD,1),RI(LD,LD,1),XR(LD,LD),XI(LD,LD)
      INTEGER N,LD,NPTS,JOB
      COMPLEX C1WRK(1),C2WRK(1)
C=====================================================================+
C   Author: Wm.Bennett/SEI              revised:  18-MAR-1986
C Purpose:  To compute the matrix residual for the Davis' Algorithm
C       JOB=0   - compute matrix residuals as follows
C                          T
C                 R := F H F  - I
C       JOB!=0  - compute matrix residuals as follows
C                        T
C                 R := F H F  - I
C=====================================================================+
      DO 20 NW=1,NPTS

       IF (JOB .EQ. 0) THEN
      !          T
      ! X := H*F
       CALL CMTX(HR(1,1,NW),HI(1,1,NW),LD,FR(1,1,NW),FI(1,1,NW),LD,
     &     XR,XI,LD,N,N,N,C1WRK,C2WRK,-1)
      ! R := F *X
       CALL CMTX(FR(1,1,NW),FI(1,1,NW),LD,XR,XI,LD,
     &     RR(1,1,NW),RI(1,1,NW),LD,N,N,N,C1WRK,C2WRK,0)

      ELSE

      ! X := H*F note that H is real only on the diagonal
       CALL CMTX(HR(1,1,NW),HI(1,1,NW),LD,FR(1,1,NW),FI(1,1,NW),LD,
     &     XR,XI,LD,N,N,N,C1WRK,C2WRK,0)
      !        T
      ! R := F *X
       CALL CMTX(FR(1,1,NW),FI(1,1,NW),LD,XR,XI,LD,
     &     RR(1,1,NW),RI(1,1,NW),LD,N,N,N,C1WRK,C2WRK,1)
      ENDIF

C       DO 5 K=1,N   ! R := X
C        CALL SCOPY(N,XR(1,K),1,RR(1,K,NW),1)
C        CALL SCOPY(N,XI(1,K),1,RI(1,K,NW),1)
C5      CONTINUE
      ! R := R - I
      DO 10 II=1,N
10      RR(II,II,NW)=RR(II,II,NW)-1
20    CONTINUE
      RETURN
      END
```

```
      REAL FUNCTION RFUNC1(ABC,LC1,LC2,LC3,KBYT,INC,npts)
C ***   INSERT COMMON BLOCK FOR STACK HERE ***
      INCLUDE 'SPEC_STACK.inc'
      INTEGER KABC,KORD,NDIM,INC1
      DATA KABC,KORD,NDIM,INC1/0,0,0,0/
C=====================================================================
C       author:  Wm.Bennett/SEI                    Revised: 30-Jan-1986
C  Purpose: to evaluate a real valued function by interpolation of data
C contained in a stack.
C  Variables:
C       LC1     - integer absolute location of common block
C       LC2     - integer absolute location of abcaissa in common block
C       LC3     - integer absolute location of ordinate in common block
C       KBYT    - integer size (in bytes) of varialbes
C                 (4 bytes for REAL*4)
C       INC     - integer increment for noncontiguous storage for ordinate
C       NPTS    - number of data points to use in abcissa/ordinate
C=====================================================================
      RFUNC1 = TRi1(STKR(KABC),STKR(KORD),INC1,NDIM,ABC)
      RETURN
C     * * * enter internal pointer/data by extended argument list
      ENTRY RFIN1(ABC,LC1,LC2,LC3,KBYT,INC,NPTS)
       KABC = (LC2-LC1)/KBYT +1
       KORD = (LC3-LC1)/KBYT +1
       NDIM=NPTS
       INC1=INC
       RFIN1=0.0
      RETURN
      END
```

```
      Subroutine Specin2(Hrel,W,Inc,Npts,Frel,Fimg,Irow,Jcol,
     $     Alim,Epsrel,Epsabs,Limit,Lenw,Iwork,Work,Ier,Neval)
C-----------------------------------------------------------------------
C     Purpose:
C     To Initialize The Reciprocal Anticausal Spectral Factor For The
C     Transfer Function Data Contained In The Array Hrel (H Is Real Valued)
C
C     External Variables:
C     Hrel              - The Real Part Of The Transfer Function
C
C     Frequency Response Data:
C     W                 - Array Of Imaginary Frequency Data Points
C     Inc               - Increment Integer For Nonsequential Storage Of
C     .                   Data (Inc=1 If Contiguous Storage)
C     Npts              - Number Of Data Points To Be Processed
C     Epsrel,Epsabs,    - Relative/Absolute Precision For Quadrature
C
C     On Return:
C     Frel, Fimg        - The Real/Imag Parts Of The Reciprocal Anticausal
C     .                   Spectral Factor
C
C     External Refs:
C     Ah2               - External Function Returns The Real Part Of H(W)
C-----------------------------------------------------------------------
      External Ah2
      Real*4  Hrel(1),Frel(1),Fimg(1),W(1)
      Integer Iwork(1),Neval,Nev
      Real*4  Work(1),Answer
      Complex Cx
C--
C--   The Program
C--
      Pi = 4.*Atan(1.0)
      Tmp = Ahin2(W,Irow,Jcol,0)
      neval=0
      Do K = 1,Npts
C--
C--   Compute The Hilbert Transform Of The Log(H)
C--
         Kk = (K-1)*Inc+1
         Call Qawc(Ah2,-Alim,Alim,W(K),Epsabs,Epsrel,
     $        Answer,Abserr,Nev,Ier,
     $        Limit,Lenw,Last,Iwork,Work)
         type *,answer,nev,ier
         Type *,'W = ',W(K),', Error = ',Abserr
         Neval=Neval+Nev
         Hiltrs = Answer/Pi
         Cx = Cexp(Cmplx(0.,-Hiltrs/2.))
         Cx = 1/(Sqrt(Hrel(Kk))*Cx)
         Frel(Kk) = Real(Cx)
         Fimg(Kk) = Aimag(Cx)
         Open(Unit=20,File='Intermediate.results',
     $        Shared,Access='Append',
     $        Form='Unformatted',Status='Old')
         Write(20) W(K),Hrel(Kk),Frel(Kk),Fimg(Kk)
         Close(20)
      End Do
      Return
      End
```

```
      SUBROUTINE STOC(N,SXR,SXI,INCX,CX,INC)
C
C       COPIES REAL/IMAGINARY PARTS OF  A VECTOR INTO COMPLEX ARRAY
C
      REAL SXR(1),SXI(1)
      COMPLEX CX(1)
      INTEGER INC,INCX

      IF(N .LE.0) RETURN
      IF (INCX.EQ.1.AND.INC.EQ.1) GOTO 20
C  . . .  INCREMENTS NOTE EQUAL TO ONE
      IX =1
      IC = 1
      IF(INCX.LT.0)IX=(-N+1)*INCX + 1
      IF(INC.LT.0) IC=(-N+1)*INC + 1
      DO 10 I=1,N
        CX(IC) = CMPLX(SXR(IX),SXI(IX))
        IX = IX + INCX
        IC = IC + INC
10    CONTINUE
      RETURN
C  . . .  CODE FOR BOTH INCREMENTS = 1
20    M=MOD(N,7)
      IF(M .EQ. 0) GOTO 40
      DO 30 II=1,M
30      CX(II)=CMPLX(SXR(II),SXI(II))
      IF(N .LT. 7) RETURN
40    MP1 = M+1
      DO 50 I=MP1,N,7
        CX(I) = CMPLX(SXR(I),SXI(I))
        CX(I+1) = CMPLX(SXR(I+1),SXI(I+1))
        CX(I+2) = CMPLX(SXR(I+2),SXI(I+2))
        CX(I+3) = CMPLX(SXR(I+3),SXI(I+3))
        CX(I+4) = CMPLX(SXR(I+4),SXI(I+4))
        CX(I+5) = CMPLX(SXR(I+5),SXI(I+5))
        CX(I+6) = CMPLX(SXR(I+6),SXI(I+6))
50    CONTINUE
      RETURN
      END
```

```
      FUNCTION TRI1(X,Y,IC1,N,XB)
C-----------------------------------------------------------------------
C Interpolation by linear splines.
C Uses a nonsequential array storage for the data points for the ordinate ONLY!
C  Variables:
C       IC1     - increment for noncontiguous memory storage for Y array
C-----------------------------------------------------------------------
      DIMENSION X(1),Y(1)
      IF(XB .GT. X(N)) THEN
       TRI1 = Y((N-1)*IC1+1)
       RETURN
      END IF
      IF (XB.LT.X(1)) THEN
       TRI1 = Y(1)
       RETURN
      END IF
      DO 2 J=1,N
C                 ---------------------------------------------------
C     try to use an equal number of given points to
C     the left and right of the interpolation point, XB
C                 ---------------------------------------------------
      IF(X(J)-XB)2,1,3
C                 ---------------------------------------------------
C     if the interpolation point is a given point, use
C     the given function value and exit
C                 ---------------------------------------------------
    1 TRI1=Y((J-1)*IC1+1)
      RETURN
    2 CONTINUE
      J=N
    3 continue
       XM = (Y((J-1)*IC1+1) - Y((J-2)*IC1+1))/(X(J)-X(J-1))
      TRI1 = XM*(XB-X(J-1)) + Y((J-2)*IC1+1)
      RETURN
      END
```

END

5-87

DTIC